Practice quiz 1 solutions

This practice quiz is intended to help you study for the real quiz. However, it varies from the real quiz in a number of ways:

- It's much longer
- "It has a number of essay questions, which take too much time for an in-class quiz. So the real quiz will focus more on having you write small amounts of code as well as showing you code and asking you questions about it, e.g. how does it work, or why does it do this broken thing and how do you fix it.
- The real quiz will include a glossary of Unity stuff so you won't have to memorize methods or fields.

Question 1

List three problems with Unity's serialization mechanism

Answer:

- Doesn't serialize references except for certain built-in classes. So if you have two pointers to the same object, they will get serialized (and then reloaded) as separate objects
- Doesn't handle null references except in certain cases. So public variables that are set to null will always reload as objects with their default values
- Doesn't handle subclassing/polymorphism properly. If you have a field of type Parent, which has subclass Child, the field will be serialized as if it the object is a Parent, even when it's a Child.
- Doesn't handle generic types (e.g. Dictionary<T1, T2>) except in very specific cases (List<T>).

Question 2

If your game is written with a straight object-oriented style, then you can test whether an object is of a given type using the **is** operator in C#. But in Unity, all your game objects are of one type (GameObject). How do you determine whether a given game object is a specific kind of game object (e.g. one representing a player)?

Answer: you check to see if it contains a component that implements the functionality of a player. For example, in the tank game, you would call GetComponent<TankController> on it to see if it has a TankController. If it is, it's a tank, and if it isn't, then it's not a tank.

Question 3

Your friend wrote a game in Unity. Here's their code:

```
void Update(GameObject o) {
   transform.position += transform.position + velocity*Time.deltaTime;
}
```

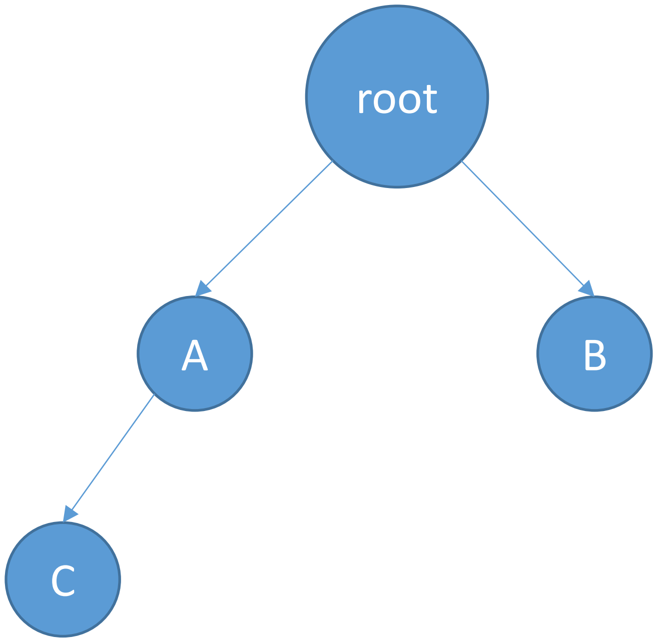
It compiles fine, but the object never moves. Explain why.

Answer:

Unity only recognizes the Update procedure if it takes no arguments, has exactly the name "Update", and returns void. Otherwise, it just assumes it's an unrelated procedure and never calls it.

Question 4

Suppose you have some game objects in the scene hierarchy (Unity's version of a scene graph):



Where root is the root of the hierarchy, and A, B, and C are GameObjects with local transform matrices T_A, T_B, and T_C, respectively, i.e. T_X translates X's local coordinate system into the coordinate system of its parent. What is the transform matrix from C to the world coordinate system (i.e. the coordinate system of the root)?

Answer: $T_A T_C$

Question 5

You're writing a soccer game. You want all the fan NPCs (non-player characters) to go crazy when there's a goal, and the AI programmer has written a component called FanNPC for those characters that has a method called GoCrazy() that will make the character dance around. You've added a trigger to the GameObject for the goal, and you'd added a script component to the goal with an OnTriggerEnter2D method to notice when a ball enters the goal:

```
void OnTriggerEnter2D(Collider2D other) {
    if (other.name == "the ball")  // Not great style, but makes the problem simpler
        ???
}
```

The problem is that it's not the goal that you want to know about the ball, it's all the little fan NPCs. What do you write in the ??? area to call the GoCrazy() method on all the fan NPCs?

Answer:

```
foreach (var fan on FindObjectsOfType<FanNPC>())
   fan.GoCrazy();
```

Question 6

In object-oriented programming, if you wanted to have a certain kind of thing in your game, such as a particular kind of enemy, you might do it by making a new subclass of game objects, and add code to its constructor to assign the proper values to its fields. However, in games with component architectures and serialization (such as Unity), you use a different mechanism? What is that mechanism?

Answer:

You'd make a prefab containing all the components needed for that kind of game object, and configure their field values appropriately within the editor. Then when you need to make one, you would instantiate the prefab.

Question 7

You write some code and it gets a compiler warning on the following line:

```
float x = Math.Sin(theta)+y*0.5;
```

And says that you can't implicitly convert a double to a float. What's wrong, and how do you fix it?

Answer: The right-hand side value is a double because Math.Sin() returns a double and 0.5 always means a double. The compiler always complains when you try to store a double in a float because you'll lose precision. The best fix would be to change the Math.Sin to Mathf.Sin (returns a float) and change 0.5 to 0.5f, so that the computations are being done in single-precision to begin with. But you could also just cast the result to float

Question 8

Explain why game engines such as Unity do everything with 4×4 matrices even when they're only operating in 3D.

Answer: They use projective coordinates (i.e. using a 4-vector to represent 3-space) because it allows them to represent translations and other transformations as linear functions (i.e. matrix multiplications).

Question 9

Suppose you were implementing a particle simulation. Would you use Euler or Verlet? Why?

Answer: Verlet, because it's much more accurate and therefore more stable.

Question 10

Suppose you build a game with a rock/paper/scissors mechanic with four choices. The win/lose table for the mechanic looks like this (the cells show the winner in each case):

	A	**B**	**C**	**D**

A	Draw	A	C	D
B	A	Draw	B	B
C	C	B	Draw	D
D	D	B	D	Draw

This is a bad game design. Explain why.

Answer: No one would ever play A or C, since they each only win in 1 out of 4 scenarios, whereas B and D win in 2 out of 4. But if nobody plays A or C (meaning people only play B and D), then you should always play B, because D always loses to B. Admittedly, this would lead to a possibility of people playing the strategy of playing A, on the assumption that people would play B. But it's still broken as a RPS mechanic, since the different strategies have different levels of strength.

Question 11

Suppose you're writing a game in Unity and it has a component class, ScoreManager, which keeps track of scores. You know there will only ever be one score manager in the game, and you need to write some code that finds it and calls a method on it. What's the simplest way to do that?

Answer: The simplest thing to do is to call FindObjectOfType<ScoreManager>(). However, there are other things you could do too. You could put it in a GameObject with a special name and then look that object up by name. Or you could add a field to every component that needs to access the score manager to hold a pointer to the score manager and manually drag the score manager into that field in every object. But life is easier if you just call FindObjectOfType.

Question 12

A. What IDE do you use with Unity?
   No solution given here (depends on what you)

B. Explain how to use it to set a breakpoint in your game.
   No solution given here (depends on your IDE)

Question 13

Explain what a material is in computer graphics.

Answer: A material specifies how the GPU chooses colors for the pixels it draws. It consists roughly of a GPU program (the pixel shader) together with a collection of parameters for it such as texture map and color tint.

Question 14

Many rendering engines render transparent objects separately from other (opaque) objects. That is, they draw all the opaque objects first, and then the transparent ones. Explain why. Is there special processing that needs to be done when drawing the transparent objects that isn't necessary for opaque ones?

Answer: This is because the alpha blending operation isn't commutative – drawing red on top of blue gives a different result than drawing blue on top of red. So the opaque objects must be drawn first, and then the transparent objects are drawn afterward, in decreasing order of distance from the camera. This is known as the painter's algorithm, and requires sorting the objects, or ideally the triangles themselves, by depth.

Question 15

Explain why stiff springs are a problem for physics engines.

Answer: For any given stiffness, there is a maximum Δt beyond which the update equations for physics integration become unstable. When that happens, the stretch on the spring grows larger over time, not smaller, leading to an increase in energy in the system and eventually the system "blows up", i.e. objects start moving pathologically fast. Conversely, for any give Δt, there's a maximum spring stiffness the system can tolerate. So very stiff springs are problematic because they may require impractically small update intervals.