1. **Which of the following would successfully evaluate to a value of -1 when the gamepad stick is fully left?**
2. CrossPlatformInputManager.GetButton(“Horizontal”);
3. CrossPlatformInputManager.GetButton(“Vertical”);
4. CrossPlatformInputManager.GetAxis(“Horizontal”);
5. CrossPlatformInputManager.GetAxis(“Vertical”);
6. **Given the following parameters, what would the value of xOffset be?**

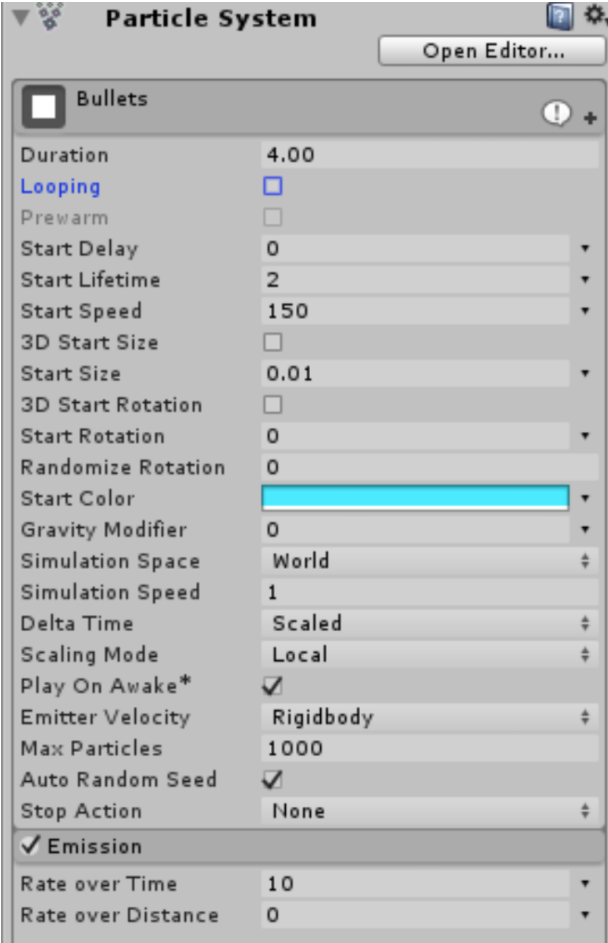
xSpeed = 12 ms^-1

Frame Rate = 60 FPS

Control throw = +0.5

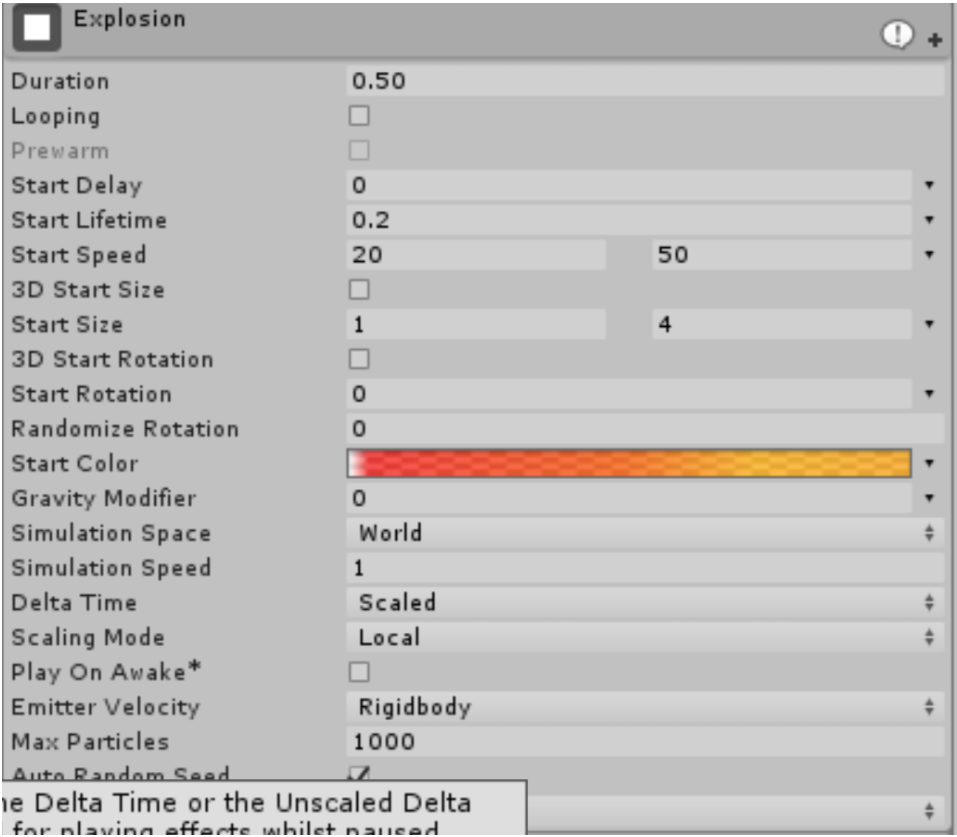
1. 0.1 m
2. 10 m
3. 7.2 m
4. 0.05 m
5. **What does the following expression evaluate to?**

**Mathf.Clamp(7f, 6f, 11f);**

1. 1
2. 7
3. 6
4. 11
5. **If an aeroplane's nose raises, as in when you rotate on take off, what's this called?**
6. Scary!
7. Roll
8. Pitch
9. Yaw
10. **Which line of code would successfully set a game object's local rotation to x = 0, y= -20, z = 30 as shown in the inspector?**
11. localRotation = Quaternion.Euler(0f, -20f, 30f);
12. transform.localRotation = Quaternion.Euler(0f, -20f, 30f);
13. transform.localRotation =Euler(0f, -20f, 30f);
14. transform.localRotation = Quaternion.Euler(-20f, 30f);
15. **In the following code, what is the correct set of couplings?**
16. **float pitchDueToPosition = transform.localPosition.y \* positionPitchFactor;**
17. **float pitchDueToControlThrow = yThrow \* controlPitchFactor;**
18. **float pitch = pitchDueToPosition + pitchDueToControlThrow;**
20. **float yaw = transform.localPosition.x \* positionYawFactor;**
22. **float roll = xThrow \* controlRollFactor;**
24. **transform.localRotation = Quaternion.Euler(pitch, yaw, roll);**
25. Pitch to position and control throw  
    Yaw to position  
    Roll to control throw
26. Pitch to position   
    Yaw to position and rotation  
    Roll to control throw
27. Pitch to position and control throw  
    Yaw to rotation  
    Roll to control throw
28. Pitch to position and control throw  
    Yaw to control throw  
    Roll to position
29. **When should you take the time to tune your gameplay?**
30. At the end of your project, after all bugs have been fixed.
31. Any time that you think your game “doesn’t feel right”.
32. Each day – your game should always have the most up-to-date tuning possible.
33. Only after a proper playtest session with players.
34. **How many particles will be emitted from this particle system?**
35. 40
36. 8
37. 300
38. 20

**Yes, Duration (4) \* Emission Rate Over Time (10) = 40**

1. **How fast will the very first particle emitted from this system be?**



1. 20 units of speed
2. 70 units of speed
3. Somewhere between 20 and 50 units of speed
4. 0.50 units of speed
5. **When is a good time to get feedback from other people on your game?**
   1. After ALL the features have been implemented.
   2. After each new feature has been implemented.
   3. After the game is fully tuned and polished
   4. Never – what do other people know?