Game design and development
Practice quiz 3 Solutions

Question 1

Suppose you run the following loop:

```
for (float i=0.0; i<100000000.0; i=i+1.0)
   do something
```

Yes, we know this is a stupid thing to do. Don't worry about that. But suppose you run it and it never terminates no longer how long you run it. Explain why. Assume standard IEEE floating point with a 24 bit mantissa.

Answer: Floating-point round-off error. With a 24 bit mantissa, the largest integer you can represent accurately is around 16 million. After that, 1 is less than the round-off error of the representation and so adding 1 doesn't change the value of i. So it gets to $2^{24} + 1$ and then stops updating. Not that we asked for this question, but you could fix it be either switching i either to double precision or to be a 32 bit int.

Question 2

Your friend decides that SQT representations are too much work and they write their animation system to represent joint poses using raw matrices. They then linearly interpolate the matrices between keyframes. Their characters appear to bend and deform horribly when they move. Explain why.

Answer: While the individual matrices may represent valid rigid body transformations (i.e. a rotation + translation), the average of two such matrices is not in general a rigid body transform. So more specifically, if you average two rotation matrices, you won't generally get a rotation matrix, you'll get something that's a linear transform, but not a valid rotation.

Question 3

Your friend was building a game and wanted to have two characters hold hands. So he attached their hands using an invisible, zero-length spring and set its stiffness to 10^8. The characters exploded. Explain why.

Answer: High stiffness springs are numerically unstable. The stiffness of 10^8 means that if the hands are pulled 1mm apart (i.e. 10^{-3}), it generates a spring force of 10^5, which likely moves them more than 1mm, leaving them father apart on the next frame, producing a larger force, etc.

Question 4

Suppose you have a data structure to represent a bounding sphere for collision detection:

```
class BoundingSphere {
   public Vector3 Center;
   public float Radius;
}
```

Write a procedure: bool Intersects(BoundingSphere a, BoundingSphere b), that returns true when the two spheres intersect. (Note: pseudocode is fine for this; don't worry about the details of C# syntax if you don't want to). Assume that you have available subroutines for whatever linear algebra you want to do (dot products, cross products, distance between vectors, matrix multiplication, etc.)

Answer:

```
bool Intersects(a, b) {
  return Vector3.Distance(a.Center, b.Center) < a.Radius+b.Radius;
}
```

using DistanceSquared is fine too, of course.

Question 5

Suppose you need to make a balloon that pumps up and pops. When a method Go() is called, the balloon starts growing, so that after 1 second, it's twice as big as the original, after 2 seconds, it's 3 times as big as the original, after 3 seconds, it's 4 times as big, etc. Write the code for a component class Balloon that contains Go() and Update() methods to make the balloon grow for 10 seconds and then destroy itself. You may add whatever fields to the class you like.

Answer:

```
class Balloon : MonoBehaviour {
  float startTime = -1;
  public void Go() { startTime = Time.time; }
  public void Update() {
     if (startTime > -1) {
       var delta = Time.Time-startTime;
       if (delta>=10) Destroy(gameObject);
       transform.scale = new Vector3(delta+1, delta+1, delta+1);
     }
  }
}
```

Question 7

Suppose q_0 is a quaternion representing some rotation about the X axis by some angle, θ_X, and q_1 is a quaternion representing a rotation around the Y axis by some angle θ_Y. Give the expressions for the quaternions representing:

1. The rotation around X followed by the rotation around Y?
2. The rotation around Y followed by the rotation around X?
3. The rotation around X by an angle $\theta_X/2$?
4. The rotation halfway between q_0 and q_1. That is, suppose an object starts out at the rotation represented by q_0 and smoothly rotates to the rotation represented by q_1. What is the rotation it is in when it is halfway from one to the other?

Answers:

1. $\boldsymbol{q_1 q_0}$
2. $\boldsymbol{q_0 q_1}$
3. $\boldsymbol{q_0^{1/2}}$**, i.e.** $\boldsymbol{\sqrt{q_0}}$
4. $\boldsymbol{\left(q_1 q_0^{-1}\right)^{\frac{1}{2}} q_0}$