

Questions

Sweet! Here are **a few 3D math-flavored Python coding interview questions** tailored for tech art — with increasing difficulty. I can go over answers/solutions with you too whenever you're ready.

☐☐ Easy: Vector Basics

1. Find the direction vector between two points. Normalize it.

```
# Given:
point_a = [3, 2, 1]
point_b = [6, 4, 5]

# Write a function:
# get_normalized_direction(a, b) -> unit direction vector from a to b
```

☐☐ Medium: Dot & Cross Product Logic

2. Determine if an object is facing a target.

```
# Given:
# - forward: the normalized forward vector of an object
# - to_target: vector from object to target position

# Write a function:
# is_facing(forward, to_target) -> True if angle between < 45 degrees
```

3. Create an orthonormal basis given one direction vector.

```
# Given:
# - a normalized forward vector

# Write a function:
```

```
# generate_basis(forward) -> (right, up)
# Such that forward, right, up are all orthogonal and normalized
```

☐☐ Hard: Matrix Transformations

4. Apply a transformation matrix to a 3D point.

```
# Given:
# - a 4x4 matrix (numpy array)
# - a 3D point [x, y, z]

# Write a function:
# transform_point(matrix, point) -> transformed 3D point
# Use homogeneous coordinates internally
```

5. World to Local Conversion

```
# Given:
# - world_matrix of an object
# - world position of a point

# Write a function:
# convert_to_local(world_matrix, world_point) -> point in object's local space
```

☐☐ Bonus: Rotation Math (Quaternions)

6. Interpolate between two rotations using quaternions.

```
# Given:
# - rotation_a and rotation_b as Euler angles
# - a blend value t in [0, 1]

# Write a function:
# blend_rotations(a, b, t) -> resulting Euler rotation
```

Let me know which one you want to tackle first, and I can walk through it with you or just check your answer!

Revision #1

Created 3 April 2025 22:46:19 by victor

Updated 3 April 2025 22:46:26 by victor