Four stages of Rendering Pipeline

1. Application Stage : Making the object, handles user input, and collision detection. Done mostly on CPU
2. Geometry Processing :
   1. Model space > World space > View space
   2. Vertex Shader : move vertices from model space to view space
   3. Projection : moves vertices to unit volume, perspective or orthographic
   4. Clipping : cut the parts of vertices outside the unit volume
   5. Screen Mapping : Map the vertices to the screen
3. Rasterization : from 2d vertices, make fragment. Check if a pixel is inside a triangle or not
4. Pixel Processing : Find color of each pixel. We should use Z-buffering to determine each fragment is used or not.

Explain z-buffering in pixel processing stage and its necessity.

Z-Buffering determines which fragment should be used to find the color of a pixel. We store the distance between the image place each fragment in z-buffer and use the fragment which is the closest to the pixel. z-buffering should be one in pixel processing because each fragment is processed in a parallel way before pixel processing.

Compare CPU and GPU in a way they handle latency. Keep in mind that latency (or system stall) is generally caused by data transfer.

Data transmission causes latency. CPU handles latency by having cache and transmit the data used often to the cache beforehand.

GPU handles latency by processing the data in a parallelized way. It also handles latency by switching the instructions when the data transmission is needed.

Explain tessellation and geometry shader and their necessities.

Tessellation: generates more vertices to represent smooth surface

Geometry shader: turn a primitive into other primitives.

They are needed to express LoD(level of detail). They are also needed to reduce the

amount of data transferred from CPU to GPU.

Explain why the last (or the fourth) row vector in any affine transform should be (0 0 0 1).

After affine transform, parallel lines should be parallel. Parallel lines are met at the point at infinity. If the last row vector is not (0 0 0 1), points at infinity may move to the finite region. Therefore, the last row vector should be (0 0 0 1)

VBO and VAO

VBO (Vertex Buffer Object) is a data object that represents storage for vertex attributes. It is used to reduce size of data transferred from CPU to GPU. It is done by 3 steps. 1) Generate new Buffer Object. 2) Bind the buffer to specific type 3) Copy vertex data to buffer object.

VAO (Vertex Array Object) is an internal data object of OpenGL that holds reference to buffers (ex. VBO) that is associated with vertex attributes. It does not copy the contents of the buffers. It is used to reduce the computation time.

Three components of camera

1. Center : Aim of camera.
2. Eye : Center position of camera.
3. Up vector : Vector that determines “up” of the camera.

Three types of Camera tools

1. Dolly Tool : Zoom in or out. Only effective on perspective. Only the eye of camera is affected.
2. Tumble Tool : Move camera’s azimuth and elevation to see different aspects of the object. Only the center of the camera is fixed.
3. Track Tool : Move camera’s position by camera’s left/right/up/down. Only the up-vector is fixed.

How to calculate up-vector

1. Calculate direction vector using the difference between two points.
2. Cross product of y axis and direction vector to get right vector.
3. Cross product between right vector and direction vector to get up-vector.