2D

GRAPHICS & JAVA

1. What things do we need to render graphics in Java?
2. Where should we preload images, and why should we do it?
3. What is **animation**?
4. What is a **sprite**?
5. How do we implement animation in Java?
6. What are some key components of the **animation** class?
7. What is **flicker**, and why does it happen?
8. What is **double buffering**? How does it work?
9. What is **page flipping**? How does it work?
10. What is **tearing**, and why does it happen?
11. Describe some strategies that can be used to avoid using **too much memory** in a 2D game.

SPRITES & TRANSFORMS

1. Why should we keep **draw()** out of the update method?
2. Name some types of **image transforms** we can do.
3. What **class** can be used to perform image transforms?
4. What are the **3 main properties** of a sprite?
5. Describe the process of using this class.

USER INTERFACE – EVENTS

1. What is an **event listener**?
2. Give some **examples** of event listeners.
3. Describe the **purpose** and **components** of the GameCore class.

PLATFORM GAMES

1. Describe the process for drawing to the game world.
2. What is a **tilemap**?
3. Write an example of a **tilemap file**.

DRAWING SPRITES

1. Why is it best to do background images as **sprites** rather than images?
2. What is **sectioning**, and how does it work?
3. How can **ordered/unordered lists** be used to draw sprites and objects to the screen?
4. Describe some methods we can use for **collision detection** between a sprite and the tilemap.
5. How can we detect **collision direction**?
6. Describe some issues with determining **when** and **where** two elements have collided.
7. Describe the general **animation loop** of a 2D game.

SOUND

1. Describe the **5 main components** of playing a sound in Java.
2. What is a **sound filter**?
3. Give some examples of **sound filters**.
4. Write pseudocode for a **simple sound filter**.

5. What is meant by **big endian** and **little endian**?
6. Describe the process of using a **SoundFilter** class.
7. Why should we play sounds in **separate threads**?
8. Give some examples of **uncompressed**, **compressed**, **lossy compressed** and **music notation** formats of music.
9. How does **MIDI** work?
10. Why might we want to use MIDI? Why not?

3D

3D GRAPHICS OBJECTS

1. What **two main things** do we need to create 3D scenes?
2. Describe the 2 main **coordinate systems** in 3D programming.
3. What is a **view window**?
4. What is the **view frustrum**?
5. Describe the process of **projecting a point onto the screen** along with the **formula** for doing so.

3D GRAPHICS PROGRAMMING

1. What is the difference between the **2D** and **3D graphics pipeline**?
2. How is a 3D object typically **defined**?
3. What components determine the **colour** of a vertex?
4. How are **non-vertex colours** determined?
5. Name some common **3D graphics libraries**.
6. Compare the power of a typical GPU with a typical CPU in terms of **processing power and speed**.
7. Name some of the **low**, **intermediate**, and **high-level** parts of 3D software.
8. What is a **scene graph**? What does it specify?
9. What part of Java3D handles the **low-level details** of drawing a 3D scene graph?
10. Draw an example of a **Java3D scene graph** (not including the view branch graph)
11. Understand all the **components** of said graph.
12. Name some **components of Java3D**.
13. What is an **alpha object**?
14. What is the **Java3D rendering loop**?

JAVA 3D SCENE GRAPH

Some of these questions overlap slightly, but I feel it is important to practice.

1. What is a **scene graph**? What does it specify?
2. Describe the **structure** of a Java3D scene graph.
3. What are the 2 main components of the **tree structure**?
4. Describe some of the **rules** for an arc.
5. Draw and describe each **component** of a scene graph.
6. What is featured in the **content branch graph**?
7. Describe some **illegal features** of a scene graph that you could have.
8. How does **compiling** a scene graph work?
9. What is a **capability**?
10. Draw an example of a scene graph for the **sphere practical**.

INTERACTION & ANIMATION IN JAVA 3D

1. What is the difference between an **interaction** and an **animation**?
2. Do both **interaction** and **animation** cause changes to the scene graph?
3. What is a **behaviour**?
4. Describe some **inbuilt behaviours**.
5. Describe the process of writing a **behaviour class**.
6. How can we use a **behaviour**?
7. How do we **add a behaviour** to a scene graph?
8. Draw an example of how a behaviour could be **represented** in a scene graph.
9. Describe some examples of **wakeup conditions**.
10. Describe some examples of **behaviour utility classes**.

ANIMATION IN JAVA 3D

1. What is **time-based animation**?
2. What is **collision detection**?
3. How do we use an **alpha object**?
4. Describe how an **interpolator** works.
5. What is **hand-crafted animation**?
6. How does **collision detection** work in Java 3D?
7. Describe the process of **implementing collision detection** in Java 3D.

ADVANCED ANIMATION

1. What is **billboarding**?
2. How does it work, and how do we use it in Java 3D?
3. What is the default **rotation** for billboarding? What other ones can we have?
4. Describe **Level of Detail** (LOD).
5. How is LOD **implemented** in Java 3D?
6. Draw a scene graph of a typical **DistanceLOD**.
7. Describe a **switch object** and its uses.
8. How is it **implemented**?
9. What is **picking**?
10. Describe some **possible uses** of picking.
11. What should be **returned** by the picking animation?
12. Describe some **features** of the pick utility class.

3D OBJECT MODELLING

1. What is a **polygon**?
2. What is a **polyhedron**?
3. Why should polygons and polyhedrons be **convex** instead of **concave**?
4. How many **sides** can a polygon have at max in Java 3D?
5. Describe the process of **hidden square removal**.
6. Describe the process of **back face culling**.
7. Describe the **painter's** and **reverse painter's algorithm**.
8. What is **z-buffering**?

JAVA 3D OBJECT MODELLING – GEOMETRY

1. Describe the 3 main ways to create **physical objects** in Java 3D.
2. What is the **purpose** of the Shape3D class?

3. Draw a scene graph showing the **typical components of a Shape3D object**.
4. What is a **primitive visual object**? Give examples.
5. What is a **mathematical class**? Give examples.
6. What is a **geometry class**? Give examples.

JAVA 3D OBJECT MODELLING – APPEARANCE

1. What methods can we use to **specify vertex colours** in Java3D?
2. What is an **appearance bundle**?
3. Name some examples of **appearance attribute node components**.
4. Can multiple appearance objects **share** attribute components?
5. Describe some examples of **attribute classes**.
6. What is **face culling**?
7. What is the difference between **back-face, no-face** and **front-face** culling?

LIGHTING

1. Name the components of a Java3D **lighting model**.
2. What **3** types of **light reflection** can we get from an **object**?
3. How does a **shading model** work?
4. What is the difference between **flat shading** and G**ouraud shading**?
5. Describe the **4 main types** of lights,
6. What types of lighting can we get from **materials**?
7. What is the **Phang Lighting Equation**?
8. Describe the process of **constructing** a lit scene.
9. What is the **region of influence** of a light? How is it **determined**?
10. What things can we do to **more closely approximate realistic lighting**?

TEXTURES

1. Why can't we go into **complex detail** with textures, such as bark on a tree or leaves?
2. What is **texture mapping**?
3. What is a **texel**?
4. What is a **MIP map**?
5. Describe the general process for using **textures** in Java3D.
6. Why can it be easier to map to a **sphere** or **cylinder** object than a **made-up polyhedron**?
7. How can we set up **texture coordinates**?
8. How can we map **non-vertex coordinates**? How are texels for **vertices** and **non-vertex** pixels determined?
9. What choices can we make to **speed up** the process when **texture rendering**?
10. What determines **final pixel colour**?
11. Draw a section of a scene graph showing the **appearance bundle with texture and TextureAttributes components**.