22BCG10019
Harsh Aditya.

Q.1

→ WPF developers (Windows Presentation Foundations) can extend XAML with procedural code written in an imperative language like C#. This integration occurs through the use of event handlers, commands and data biding which enables developers to add dynamic behavior and logic to their user interfaces.

XML is primarily used for defining the structure & layout of the UI, while C# handles the business logic, the development process becomes more modular and maintainable.

eg. event handlers in C# can respond to user interactions defined in XML (eg button clicks). Commands in C# can also be bound to UI, enabling more complex and reusable interactions.

Additionally, data binding allows the UI to automatically update based on changes

in data model without requiring explicit procedural code.

This combination simplifies tasks such as handling texts, events, binding data to UI elements and responding to user input. It makes the development process more efficient by documenting the visual representation ie XAML from the logic ie C# which also examines the scalability and readability of the code.

Also, developers can focus on business logic & UI design separately, improving productivity and code organization.

* Simplifications when using C# with XAML:

- Separation of concerns:-
  XAML focuses on the declarative definition of UI, while C# handles the procedural logic. This separation simplifies maintenance & enhances readability.

- Dynamic UI Behavior:
  Complex or conditional UI changes (eg. responding to runtime data) are easier to implement in C# as compared to XAML alone.

- Enhanced Reusability
  Procedural code in C# is easier to debug & test using standard debugging tools & techniques.

- Simplified debugging & Testing
  Procedural code in C# is easier to debug & test using standard debugging tools & techniques.

- Access to .NET libraries
  Developers can leverage the full power of the .NET framework, including libraries for data manipulation, communication & computation.

So, integration provides a powerful & flexible way to create modern, dynamic & user-friendly WPF applications.

Q2.2

→ Euler angles are a set of three angles used to describe the orientation of a rigid body in 3D space. They represent the successive rotations of a body about its coordinate axes. This is closely tied to plane rotation, as each rotation occurs in a specific plane defined by the coordinate axes.

*Components of Euler Angles:

1. Yaw (ψ) : Rotates about Z-axis.
2. Pitch (θ) : Rotates about Y-axis.
3. Roll (ϕ) : Rotates about X-axis.

Rotations occur in a specified sequence, & the sequence matters because rotations are not commutative.

Plane rotation in Euler Angles:

- Each rot^n in Euler angles corresponds to a rot^n in a plane.
- When rotating around an axis, the other two axes define the plane of rot^n.

* Yaw (ψ)

$$R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \text{Rotation matrix.}$$

- Rotation in XY plane
- Rotated about Z-axis (R_z)

* Pitch (θ)

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \rightarrow \text{Rotation matrix}$$

- Rotation in XZ plane
- Rotated about Y-axis (R_y)

* Roll (ϕ)

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \rightarrow \text{Rotation matrix.}$$

- Rotation in YZ plane
- Rotated about X-axis (R_x)

If we combine rotations,
we get $R = R_z(\psi) R_y(\theta) R_x(\phi)$.

* Applications:

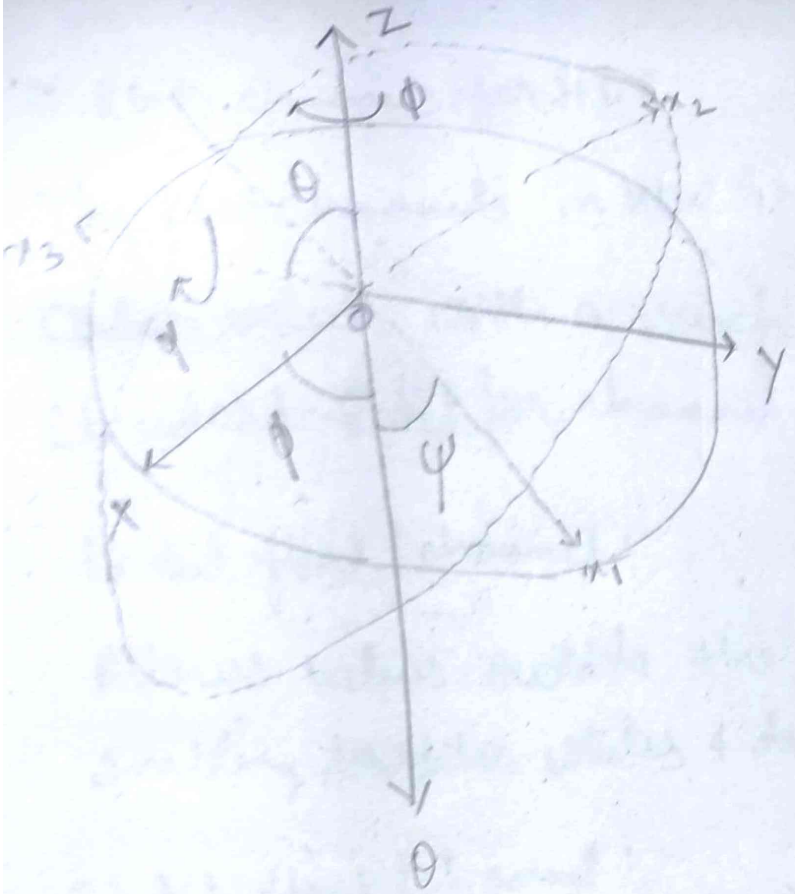
- Aerospace
  Describing orientation of aircraft & spacecrafts.

- Robotics
  Positioning robotic arms.

- Computer Graphics
  Rotating 3D objects in simulations & games.

- Physics and Engineering
  Analyzing rotational dynamics.

These angles are intuitive & widely used, but they can suffer from gimbal lock, a situation where two rotational axes align, leading to a loss of one d.o.f.



Rough sketch of Eulers angles.

Q.3

→ HDR images are designed to represent a broader range of luminance levels, capturing details in both the darkest & brightest parts of an image. In RGB format, HDR images can include additional components like depth & alpha values to enhance their representation. Here's how HDR images are represented in this format:

* RGB channels for HDR

The RGB channels in HDR images typically store color values with a much higher precision than standard 8-bit per channel formats.

16 bit float / channel:

Allows values outside the standard 0-1 range, enabling brighter whites & darker blacks.

32 bit float / channel:

Offers extremely high precision & dynamic range.

* Alpha channel

This is used to store transparency information.

In HDR images:

- Retains higher precision, to ensure smooth transition & accurate compositing.
- Transparency is useful in scenarios like layering images or rendering 3D scenes where some parts of the image may be visible.

* Depth channel

It is an optional component often used in conjunction with HDR images, particularly in 3D graphics & rendering.

- Depth values are usually normalized b/w 0 & 1 but can also use floating-point precision for greater accuracy.
- Depth information can be used for effects like depth of field, occlusion & shadows.

* File formats for HDR representation.

- OpenEXR

  Widely used in film & visual effects, supports multiple channels with 16 or 32 bit floating point precision.

- HDR (Radiance)

  Simpler format mainly for storing HDR, RGB data.

- TIFF
  Can store HDR data & additional metadata like depth.

* Combine HDR, Alpha & Depth

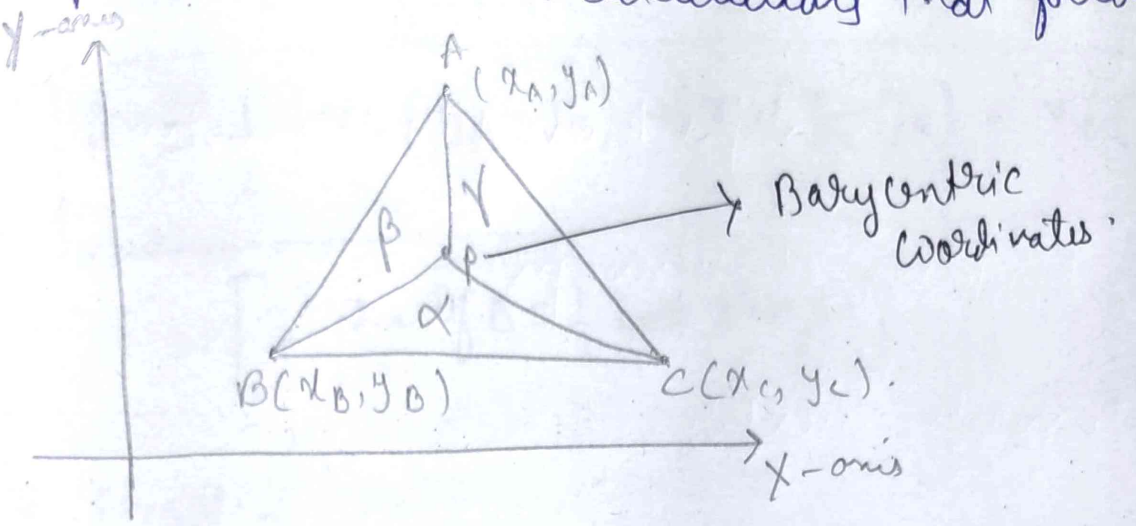
Thus, when all components are combined,

- RGB
  Stores the color & brightness information with high precision.
- Alpha
  Adds transparency for compositing & effects
- Depth
  Provides spatial information for 3D rendering & advanced post-processing effects.

Together, these channels create rich & versatile data representation suitable for modern imaging & rendering workflows.

Q. 4

→ Let us consider a diagram illustrated below & proceed with the calculations that follow:-



So, area of a triangle can be calculated using barycentric coordinates α, β, γ.

Barycentric coordinates, represent a point P inside the triangle as a weighted sum of it's vertices A, B, C.

As per barycentric coordinates,

P (inside ΔABC, point P) is expressed as

$$P = \alpha A + \beta B + \gamma C$$

- α, β, γ represent the areas of the sub triangles ΔPBC, ΔPCA, ΔPAB relative to the area of ΔABC

* Area calculation of triangle, using Barycentric coordinates.

∴ A = area of ΔABC

$$A = \frac{1}{2}\left| x_A(y_B - y_C) + x_B(y_C - y_A) + x_C(y_A - y_B) \right|$$

[∵ Area of $\Delta = \frac{1}{2}$ base × height]

* Proof:-

- Area of sub-triangles,

$$\alpha = \frac{\text{area. } \Delta PBC}{\text{area } \Delta ABC} \quad -(1)$$

$$\beta = \frac{\text{area. } \Delta PCA}{\text{area } \Delta ABC} \quad -(2)$$

$$\gamma = \frac{\text{area. } \Delta PAB}{\text{area } \Delta ABC} \quad -(3)$$

- (1) + (2) + (3) ⇒

area ΔPBC + area ΔPCA + area ΔPAB = area ΔABC.

$$\Rightarrow \boxed{\alpha + \beta + \gamma = 1} \quad -(4)$$

• ∵ Barycentric coordinates are ratios of sub triangle areas to total area of ΔABC.

thus

$$\text{area } \Delta ABC = \frac{\text{area } \Delta PBC}{\alpha} \quad -(5)$$

or,

$$\text{area } \Delta ABC = \frac{\text{area } \Delta PCA}{\beta} \quad -(6)$$

or,

$$\text{area } \Delta ABC = \frac{\text{area } \Delta PAB}{\gamma} \quad -(7)$$

ie. from (5), (6) & (7)

we get :-

$$\boxed{\text{Area } \Delta ABC = \text{Area } \Delta PBC + \text{area } \Delta PCA + \text{area } \Delta PAB}$$

Or

$$\boxed{1 = \alpha + \beta + \gamma}$$

Thus Proved.

Q.5

→ Using different views like texture & texture mapping views in computer graphics or 3D rendering provides range of benefits. These views are essential for understanding, creating, & refining the appearance of 3D objects by controlling how images (textures) are applied to their surfaces.

* Texture View:

- The texture view displays the 2D image or pattern that will be mapped onto the surface of a 3D model.
- It provides a clear understanding of the base image that will be used for the surface detailing of the object.

#Benifits of texture view.

* Editing Precision,

Designers can edit textures directly, ensuring high quality details.

- They can correct distortions or mismatches before applying the texture.

* Optimization:
  Allows for efficient resolution management to a balance quality & performance.

* Reusability:
  Textures can be stored & reused across multiple models, saving time.

* Texture Mapping View,

- It shows how the 2D texture is applied to the 3D model, based on it's UV coordinates.

- It visualized the interaction between the texture & the geometry of the model.

Benefits Of Texture Mapping Views:

* Accuracy in Placement:

Designers can adjust UV coordinates to ensure the texture aligns correctly with model's geometry.

* Realistic Rendering:

Ensures that textures look natural & realistic when rendered in 3D scenes.

* Customizable effects:

Enables designers to create effects like bump, mapping or specular highlights.

* Applications:-

- Video Games.
- Film & Animation.
- Product Design.
- Virtual Reality.

Thus, using both views allows for better control & efficiency in designing 3D assets, ultimately enhancing the scenes.