MMC 4

Friday, May 9, 2025 3:17 PM

1. Video Signal Representation

(Visual representation, Transmission, Digitalization)

- *No direct previous question found under this heading.*

2. Computer Video Format

(Introduction to Computer video format)

- **Determine the storage capacity per pixel for SVGA video controller with 1024×768 resolution, 24 bits/pixel.**

3. Computer-Based Animation

(Input process, composition stage, in-between process, changing colors)

- **Explain the computer-based animation.**
- **How can you generate animation using a computer?**

4. Animation Language

(Linear list notations, general-purpose language, graphical languages)

- **Explain the different types of animation language used in multimedia computing.**
- **Discuss the animation language.** *(Appears in some variations)*

5. Methods of Controlling Animation

(Full explicit control, procedural control, constraint-based systems, tracking live action, kinematics and dynamics)

- **What is animation? Explain the methods of controlling animation.**
- **Discuss the method of controlling animation.**

6. Display of Animation

(Basic knowledge about display of animation)

- *No direct previous question, but related questions may indirectly touch this in other topics.*

7. Transmission of Animation

(Basic knowledge about transmission of animation)

- *No direct previous question found under this heading.*

Video Signal Representation

Video signal representation is the process of displaying or transmitting motion visuals on electronic screens (TVs, monitors, etc.). It converts visual data into signals understandable by display hardware. Earlier CRT displays used electron beam scanning; modern displays use digital pixel grids.

1. Visual Representation

The goal is to create a natural and immersive viewing experience that accurately simulates real-world scenes.

i) Vertical Detail and Viewing Distance

- Determines how sharp an image appears from a specific distance.
- Closely tied to screen resolution and **aspect ratio**.

Formula:

Aspect Ratio = Width / Height

Standard Example:

- 4:3 aspect ratio (common in old TVs)
- Given: Width = 12 inches
  ⇒ Height = 12 / 1.33 ≈ **9 inches**

ii) Horizontal Detail and Picture Width

- Indicates how wide the scene appears on the screen.
- Based on aspect ratio and vertical height.

Formula:

Width = (4 / 3) × Height

Example:

- Height = 300 pixels
  ⇒ Width = (4/3) × 300 = **400 pixels**

iii) Total Detail Content of the Image

- Represents total number of pixels forming the image.
- Defines image clarity or **spatial resolution**.

Formula:

Total Pixels = Height × Width

Example:

- For 1920 × 1080 resolution:
  Total = 1920 × 1080 = **2,073,600 pixels**

iv) Perception of Depth

Depth perception simulates 3D using cues on a 2D screen.

Achieved by:

- **Perspective**: Nearer objects look bigger.
- **Lighting & Shadows**: Suggest shape and space.
- **Stereoscopic Vision**: Each eye sees a slightly different image.

Example:

3D movies use glasses to send two views (left/right eye) → brain fuses them into depth.

v) Luminance and Chrominance

- **Luminance (Y)**: Brightness content
- **Chrominance (U, V)**: Color differences

RGB → YUV Conversion (for efficient transmission):

Formulas:

Y = 0.30R + 0.59G + 0.11B
U = (B − Y) × 0.493
V = (R − Y) × 0.877

Why YUV is used:

- Human eyes are more sensitive to brightness than color.
- Reduces bandwidth.
- Used in broadcasting.

vi) Temporal Aspects of Illumination

Relates to how motion is represented over time using frames.

Two important factors:

1. **Frame Rate** – Should be high enough for smooth motion.
2. **Persistence of Vision** – Brain retains an image for ~1/16th second, making fast frames appear continuous.

vii) Continuity of Motion

- Motion appears **smooth** if enough frames are shown per second.
- **Minimum for perceived motion**: 15 fps

Typical Frame Rates:

Use Case	Frame Rate
Movies	24 fps
Television	30 fps
Gaming/Modern UI	60+ fps

Higher fps = smoother visuals

viii) Flickering

- Flicker is visible flashing due to **low refresh rate**.
- To avoid flicker, refresh rate should be **≥ 50 Hz**

Modern Display Refresh Rates:

Device Type	Refresh Rate
Basic Monitor	60 Hz
Gaming Monitor	120–144 Hz
Premium TVs	Up to 240 Hz

Higher refresh rate → less flicker, better viewing comfort.

2. Transmission of Video Signals

Video signals can be transmitted using different encoding standards. NTSC is one of the oldest analog standards, while digital systems use various encoding schemes to optimize quality and bandwidth.

NTSC Standard (National Television Systems Committee)

- Analog video transmission standard.
- Used in: **North America, Japan, South Korea**, and parts of Asia.
- Transmits:
    - **1 Luminance (Y)**: for brightness.
    - **2 Chrominance components (color information)**: encoded efficiently.

i) RGB Signal

- **RGB = Red + Green + Blue**
- Used in: Monitors, cameras, and display systems.
- Each pixel = mix of R, G, B intensities.

Limitations:

- Requires **high data bandwidth**.

- Not ideal for transmission.
- **Better for display**, not communication.

ii) YUV Signal

- Separates brightness and color for better compression.

Component	Meaning
Y	Luminance (brightness)
U	Blue - Luminance
V	Red - Luminance

Formulas:

- Y = 0.30R + 0.59G + 0.11B
- U = (B - Y) × 0.493
- V = (R - Y) × 0.877

Advantages:

- **Reduces bandwidth** (human eyes are more sensitive to brightness).
- Supports **efficient color video compression**.

Example:

TV broadcasts use **YUV** for color video while preserving clarity in brightness.

iii) YIQ Signal (Used in NTSC)

- A variant of YUV used in NTSC.
- Components:
  - **Y**: Luminance (same as YUV)
  - **I**: In-phase (hue detail)
  - **Q**: Quadrature (saturation detail)

Formulas:

- Y = 0.30R + 0.59G + 0.11B
- I = 0.60R - 0.28G - 0.32B
- Q = 0.21R - 0.52G + 0.31B

Purpose:

- Optimized for **analog TV broadcasting**.
- I and Q channels carry **fine color details**.

3. Digitalization of Video

Digitalization = Converting **analog video** into **digital format** for processing, storage, or transmission.

Steps in Digitalization

1. **Sampling:**
   - Divide video frame into an **M × N grid**.
   - Each grid point captures **brightness or color value**.
2. **Quantization:**
   - Convert sampled values into **binary numbers**.
3. **Temporal Sampling:**
   - Capture **multiple frames per second (fps)**.
   - Higher fps = smoother motion.

Example:

At **30 fps**, 30 individual images (frames) are captured per second.

Computer Video Formats & Storage Calculation

Format	Resolution	Color Depth	Storage Calculation	Size (Bytes)
CGA	320×200	2 bits/pixel	320 × 200 × 2 = 128,000 bits	128,000 ÷ 8 = 16 KB
EGA	640×350	4 bits/pixel	640 × 350 × 4 = 896,000 bits	896,000 ÷ 8 = 112 KB
VGA	640×480	8 bits/pixel	640 × 480 × 8 = 2,457,600 bits	2,457,600 ÷ 8 = 307 KB
XGA	640×480	16 bits/pixel	640 × 480 × 16 = 4,915,200 bits	4,915,200 ÷ 8 = 491 KB
SVGA	1024×768	24 bits/pixel	1024 × 768 × 24 = 18,874,368 bits	

1. **Determine the storage capacity per pixel for SVGA video controller with 1024×768 resolution, 24 bits/pixel.**

Given:

- **Resolution**: 1024 × 768 (not needed here as we only need per pixel info)
- **Color Depth**: 24 bits/pixel

Storage per pixel:

- 1 pixel = **24 bits**
- Convert to bytes:
  24 bits ÷ 8 = 3 bytes

Answer:

3 bytes per pixel
(for SVGA with 24-bit color depth)

Computer-Based Animation

Computer-based animation is the process of creating animated visuals using computers and animation software. It simulates motion by displaying sequences of images rapidly, creating the illusion of continuous movement.
This method is widely used in fields such as:

- **Entertainment** (films, games)
- **Education** (interactive learning)
- **Advertising** (animated commercials)
- **Simulation** (training modules)

Stages of Computer-Based Animation

1. Input Process (Digitizing Drawings)

This is the initial stage where physical drawings are converted into digital form.
Methods of Digitization:

- **Optical Scanning**: Scans hand-drawn images and converts them to digital format.
- **Data Tablet**: Allows artists to draw or trace directly on a tablet, capturing real-time input.

Key Frames:

- Key frames are **the major frames** that define the **start and end** of any significant movement.
- These frames represent crucial positions or poses in the animation sequence.
- Once digitized, key frames are saved for further processing.

2. Composition Stage (Combining Elements)

At this stage, all elements of the animation are assembled into a single frame.

Elements Combined:

- **Foreground Objects**: Characters, moving items, etc.
- **Background**: Static or dynamic scenery or environment.

The goal is to create a **complete visual scene** that looks coherent and integrated.

3. In-Between Process (Tweening)

Tweening refers to the generation of intermediate frames between two key frames to produce smooth motion.

Techniques Used:

- **Interpolation**: Computer automatically calculates intermediate frames.
- **Linear Interpolation**: Simplest form, where objects move in a straight line from point A to B.

This process ensures the animation appears **continuous and natural**.

Example:

- If a ball moves from **Position A** (on the ground) to **Position B** (in the air), the tweening process fills in the in-between frames showing the arc of the bounce.

4. Changing Colors (Using CLUT - Color Lookup Table)

Color changes can enhance animation effects and can be dynamically altered during animation using a **Color Lookup Table** (CLUT).

Key Concepts:

- **CLUT**: A table that maps original colors to new ones.
- **LUT Animation**: Changing values in the CLUT over time to achieve color transitions.

Example:

- If a character's outfit needs to change gradually from red to blue, the CLUT adjusts the color values frame by frame to achieve a **smooth visual transition**.

Complete Example: Ball Bouncing Animation

Stage	Description
Input Process	Initial and final positions of the ball (key frames) are digitized.
Composition	Ball (foreground) is placed on a background (floor and sky).
Tweening	Computer generates frames showing the ball's path through its bounce.
Color Change	As the ball bounces, it changes color from red to blue using CLUT.

Animation Language

Animation language refers to the formal methods or systems used to describe, create, and control animations. These languages help define **what happens**, **to which object**, **when**, and **how** during the animation sequence.

Types of Animation Language

1. Linear-List Notation

A text-based language that defines animation actions as a sequence of commands listed in order. Each command typically specifies:

- **Start Frame**: When the action begins.

- **End Frame**: When the action ends.
- **Action**: The operation (e.g., rotate, move).
- **Object**: The animated object.
- **Parameters**: Additional details like axis, angle, speed, etc.

Key Features:

- Simple and structured format.
- Suitable for scripting repetitive or time-bound actions.

Example:

45, 53, rotate, "palm", 1, 30

Interpretation:

- Start at frame 45 and end at frame 53.
- Rotate the object named "palm".
- Rotation axis is 1 (e.g., X-axis).
- Rotate by 30 degrees.

2. Graphical Language

A visual interface used to create and manipulate animations interactively. Instead of writing code, users use tools to drag, drop, and modify objects on the screen.

Key Features:

- No coding required.
- Intuitive and beginner-friendly.
- Immediate visual feedback during design.

Example Tools:

- **AutoCAD**: Enables animation through object transformations; widely used in engineering and CAD modeling.
- **Adobe Animate / Blender**: Allow detailed frame-by-frame animation using visual keyframes and timelines.

3. General-Purpose Programming Language

These are high-level programming languages used in broader software development but also capable of producing animations. They provide high control, flexibility, and integration with user interfaces.

Key Features:

- Highly flexible and programmable.
- Suitable for complex or performance-demanding animations.
- Can integrate animation with UI, logic, and data handling.

Examples:

Language	Usage in Animation
QBASIC	Simple 2D animations using loops and screen coordinates
C / C++	Used in game engines, OpenGL-based real-time graphics
Java	GUI-based animations using Swing or JavaFX libraries

Simple Example in QBASIC:

```
FOR x = 1 TO 100
  PSET (x, 100)
  SLEEP 1
NEXT x
```

.

This creates a moving point from left to right horizontally.

Methods for Controlling Animation

Controlling animation refers to how we manage the movement, timing, and interaction of animated objects. These methods determine how animations behave and respond during a sequence. Each method varies in terms of automation, realism, and control.

1. Full Explicit Control

- **Definition**: The animator manually defines every change in position, size, or orientation frame by frame.
- **Control Level**: High manual control; no automation.
- **Typical Changes Defined**:
    - Translation (movement from one place to another)
    - Rotation (spinning)
    - Scaling (resizing)

Example:
A bouncing ball is manually positioned at different heights in every frame to simulate bounce — animator defines all motion details.
Use case: Simple animations or when the animator wants complete artistic control.

2. Procedural Control

- **Definition**: Animation behavior is defined through procedures, rules, or scripts, not manually per frame.
- **Types**:
    - **Physically-based**: Objects respond based on forces and motion rules (like physics).
    - **Actor-based**: Objects (actors) send messages to each other to coordinate behavior.

Example:

- In a crowd simulation, one person (actor) walks and others adjust their movement to avoid collision.
- A bouncing ball uses gravity to fall and bounce off the floor automatically.

Use case: Games, simulations, or real-time applications where interaction and realism are needed.

3. Constraint-Based Systems

- **Definition**: Objects are animated with constraints (rules they must obey), especially when in contact with other objects.
- **Purpose**: To ensure objects move realistically together — often used for compound motions.

Example:
A character holding a briefcase — if the hand moves, the briefcase must follow. The motion of the hand constrains the motion of the object.
Use case: Character animation, robotic arm simulations.

4. Tracking Live Action

- **Definition**: Real-world object movements are recorded and used to drive the animation.
- **Tools Used**: Motion capture (MoCap) systems or video tracking software.
- **Output**: Recorded motion data is used to animate 3D characters.

Example:
An actor wears motion capture suits. Their body movements are tracked and applied to a digital character in a game or movie.
Use case: Film industry, realistic human motion in games and virtual reality.

5. Kinematics and Dynamics

a) Kinematics:

- **Definition**: Studies motion based on position, velocity, and acceleration without considering forces.
- **Type**: Describes how objects move.
- **Types**:
  - **Forward Kinematics**: Moves joints based on input angles.
  - **Inverse Kinematics (IK)**: Calculates angles needed for a body part to reach a target position.

Example:
In IK, if you want a robotic hand to touch a point, the system calculates the required angles of all joints.

b) Dynamics:

- **Definition**: Studies motion based on physical forces like gravity, friction, and mass.
- **Use**: Adds realism by simulating natural movement.
- **Calculation-heavy**, uses physics engines.

Example:
A falling object accelerates due to gravity, bounces, and then settles due to friction — this behavior is modeled using dynamics.
Use case: Realistic simulations, physics-based animation in films or games.

Display of Animation

To display animations on raster graphics systems, objects must be **scan-converted** into pixel maps (pixmaps) stored in the **frame buffer**.

Key Points:

- **Scan Conversion**: Converts object geometry into a pixel representation.
- To show **rotating or moving objects**, successive pixmaps are generated and displayed frame by frame.
- This creates the **illusion of smooth motion**.

Frame Buffer Management:

- The **frame buffer** holds image data for display.
- For animation:
  - It may be divided into **two separate image buffers**.
  - Each buffer uses **half the bits per pixel**.
  - This allows one image to be prepared while the other is being displayed (known as **double buffering**).

Transmission of Animation

Animation data can be transmitted using two main techniques:

1. Symbolic Representation

- Sends only **object descriptions** and **operation commands** (not full pixel data).
- At the **receiver's end**, scan-conversion is performed to display the animation.

Features:

- **Shorter transmission time** → due to small size of data.
- **Longer display time** → scan-conversion done after receiving data.

Transmission Time Depends On:

- Size of symbolic structures.
- Number of animated objects and operations.

Example:

- A bouncing ball in a 2D animation where the ball's position and size are described with commands (e.g., move right, bounce), and the actual pixel data for each frame is generated at the receiver's end by scanning the instructions.

2. Pixmap Representation

- Sends **full pixmap data** (raw pixel images) for each animation frame.

Features:

- **Longer transmission time** → large data size.
- **Shorter display time** → no scan-conversion required at receiver end.

Transmission Rate:

Transmission Rate = Size of Pixmap × Frame Rate

Example:

- A video file (e.g., .mp4 or .avi) where each frame contains complete pixel data for the image, and the file is transmitted over the internet. The entire video is played back at the receiver's end without any further need for computation.