SRM Institute of Science and Technology
Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204
Department of Computer Applications
Circular – 2023-24
BCA DS 6th Sem

INTRODUCTION TO ANIMATION (UDS23G04J)

Lab Manual

Lab 1: Image Manipulation in Photoshop

1a) Extract the Flower and Organize on a Background

Title: Extracting and Organizing a Flower Image

Aim: To extract a flower from a photographic image, select a background, and organize the flower on the background using Photoshop.

Procedure:

1. Open the flower image in Photoshop.
2. Use a selection tool (e.g., Magic Wand, Quick Selection) to select the flower.
3. Refine the selection if necessary.
4. Copy the flower selection.
5. Open a new or existing background image.
6. Paste the flower onto the background.
7. Resize and position the flower as desired.
8. Save the final image.

Source Code: (This is a Photoshop-based lab, so there's no traditional "source code." The procedure *is* the code, in a sense, as it outlines the steps to be followed in the software.)

Input: A photographic image of a flower, a background image.

Expected Output: A composite image with the flower extracted and placed on the chosen background.

1b) Collage Creation

Title: Creating a Collage

Aim: To design a collage using effective cropping techniques in Photoshop.

Procedure:

9. Gather several images for the collage.
10. Open the images in Photoshop.
11. Use cropping tools to select portions of each image.
12. Create a new document for the collage.
13. Copy and paste the cropped image portions into the collage document.

14. Arrange, resize, and layer the image portions to create a visually appealing composition.
15. Apply effects or adjustments as desired.
16. Save the collage.

Source Code: (Photoshop procedure)

Input: Multiple images.

Expected Output: A finished collage composed of cropped image sections.

Lab 2: Object Manipulation and Scenery Creation in Photoshop

- **2a) Object Extraction and Feathering**

  Title: Object Extraction and Feathering

  Aim: To extract objects from multiple images, organize them in a single file, and apply feather effects using Photoshop.

  Procedure:

  1. Open the images (F1.jpg, F2.jpg, F3.jpg) in Photoshop.
  2. Use selection tools to select objects from each image.
  3. Copy the selected objects.
  4. Create a new Photoshop document.
  5. Paste the objects into the new document, arranging them as desired.
  6. For each layer containing an object, apply a feather effect (Select > Modify > Feather) to soften the edges.
  7. Save the combined image.

  Source Code: (Photoshop procedure)

  Input: Images F1.jpg, F2.jpg, and F3.jpg.

  Expected Output: A single image with objects from the input images combined, with feathered edges.

- **2b) Painting a Scenery**

  Title: Painting a Park Scenery

  Aim: To paint a scenery of a park using different tools in Photoshop.

  Procedure:

  1. Create a new Photoshop document.
  2. Use the Brush tool, Paint Bucket tool, and other painting tools to create the elements of a park scene (e.g., trees, grass, paths, sky).
  3. Use different brush sizes, colors, and textures.
  4. Experiment with layering to create depth.
  5. Add details and finishing touches.
  6. Save the painted scenery.

  Source Code: (Photoshop procedure)

  Input: (None, the input is the user's creative choices within Photoshop)

  Expected Output: A digital painting of a park scene.

Lab 3: Advanced Picture Manipulation in Photoshop

3a) Picture Manipulations

Title: Advanced Picture Manipulations

Aim: To perform various picture manipulations using all possible tools of Photoshop.

Procedure:

1. Open an image in Photoshop.
2. Experiment with a variety of tools:

    Selection tools (Magic Wand, Lasso, etc.)

    Adjustment layers (Brightness/Contrast, Hue/Saturation, etc.)

    Filters (Blur, Sharpen, etc.)

    Retouching tools (Clone Stamp, Healing Brush, etc.)

    Layer masks

    Transformations (Scale, Rotate, Skew, etc.)

3. Combine multiple tools and techniques to achieve desired effects.
4. Document the manipulations performed.
5. Save the manipulated image.

Source Code: (Photoshop procedure)

Input: An image.

Expected Output: A significantly altered version of the original image, demonstrating various Photoshop manipulation techniques.

3b) Magazine Cover Modification

Title: Magazine Cover Modification

Aim: To take a magazine cover image and make changes using selection tools in Photoshop.

Procedure:

6. Obtain a digital image of a magazine cover.
7. Open the image in Photoshop.
8. Use selection tools to isolate elements of the cover (text, images, etc.).
9. Modify the selected elements:

    Change text content.

    Replace images.

Alter colors.

Rearrange layout.

10. Use other Photoshop tools as needed to blend the modifications seamlessly.
11. Save the modified magazine cover.

Source Code: (Photoshop procedure)

Input: A digital image of a magazine cover.

Expected Output: A modified version of the magazine cover with altered text, images, and/or layout.

Lab 4: Animation

4a) Growing Moon Animation

Title: Growing Moon Animation

Aim: To create an animation representing the growing moon.

Procedure:

1. Choose a software (Photoshop, Flash/Animate, or other animation software).
2. Create a series of frames showing the moon's phases, from crescent to full.
3. In each frame, draw or import an image of the moon at a slightly different stage of growth.
4. Arrange the frames in sequence.
5. Set the frame rate and playback settings.
6. Preview and refine the animation.
7. Export the animation in a suitable format (e.g., GIF, video).

Source Code: (This will vary depending on the software. For example, in Adobe Animate, it would involve ActionScript or JavaScript. In Photoshop, it would involve timeline settings and layer manipulation.)

Input: (None, the input is the user's creation of the moon phases)

Expected Output: A short animation showing the moon gradually growing in size and changing its shape.

4b) Cricket Game Simulation

Title: Cricket Game Simulation

Aim: To create a simple simulation of a cricket game.

Procedure:

8. Choose a software (e.g., Scratch, Unity, or a game development library).
9. Design the basic elements:

   A playing field.

   Batsman and bowler figures (simple shapes or sprites).

   A ball.

10. Implement basic game mechanics:

    Ball movement.

    Batsman hitting the ball (controlled by user input).

    Scoring.

11. Add simple animation:

    Ball trajectory.

    Batsman's swing.

12. Include a basic user interface (start button, score display).
13. Test and refine the simulation.
14. Package or export the simulation.

Source Code: (This will vary greatly depending on the software. Here's a *conceptual* example using a simplified structure, not actual code)

```
// Simplified conceptual code (not executable)
function initializeGame() {
  // Set up field, batsman, bowler, ball
  score = 0;
}

function handleInput(userInput) {
  // Process user input (e.g., swing bat)
}

function updateBallPosition() {
  // Calculate ball movement, collision
}

function updateScore() {
  // Calculate and display score
}

function drawGame() {
  // Draw field, players, ball, score
}

function gameLoop() {
  handleInput(userInput);
  updateBallPosition();
  updateScore();
  drawGame();
}
```

Input: User input to control the batsman (e.g., button clicks to swing).

Expected Output: A very basic, interactive simulation of a cricket game, showing ball movement, batting, and scoring.

Lab 5: Animation

5a) Ball Bouncing on Steps

Title: Ball Bouncing on Steps Animation

Aim: To create an animation showing a ball bouncing on a set of steps.

Procedure:

1. Choose animation software (e.g., Flash/Animate, After Effects, or a 2D animation library).
2. Create a visual representation of the steps (a series of descending rectangles).
3. Draw a ball.
4. Animate the ball's movement:

    Start the ball at a height.

    Make the ball fall, accelerating downwards.

    Make the ball bounce on each step, losing some energy (reduced height) with each bounce.

    Continue until the ball's bounces are very small.

5. Adjust the timing and spacing of the bounces for realistic motion.
6. Add any desired visual effects (e.g., squash and stretch of the ball on impact).
7. Export the animation.

Source Code: (Varies by software. In Animate/Flash, this would involve keyframes and motion tweens or ActionScript. A simplified conceptual example:)

```
// Simplified conceptual code
for (step = 0; step < numberOfSteps; step++) {
   // Ball falls to step
   animateBallFall(currentHeight, stepHeight[step]);
   // Ball bounces
   animateBallBounce(stepHeight[step], bounceFactor);
   currentHeight = stepHeight[step] * bounceFactor;
}
```

Input: (None, the input is the user's design of the steps and ball motion)

Expected Output: An animation of a ball realistically bouncing down a set of steps.

5b) Character Walk Animation

Title: Character Walk Animation

Aim: To create a character walk animation.

Procedure:

8. Choose animation software (e.g., Flash/Animate).
9. Design a simple character.
10. Break down the walk cycle into key poses (contact, down, pass, up).
11. Draw the character in each key pose.
12. Create in-between frames to smooth the motion.
13. Animate the character's body, arms, and legs.
14. Adjust timing and spacing for a natural walk.
15. Export the animation.

Source Code: (Primarily keyframes and tweens in animation software. May involve scripting for more complex motion.)

Input: (None, the input is the user's design of the character and walk cycle)

Expected Output: A short animation of a character walking.

Lab 6: Simulation

6a) Simulate Movement of a Cloud

Title: Cloud Movement Simulation

Aim: To simulate the movement of a cloud.

Procedure:

1. Choose animation software (e.g., Flash/Animate, or a graphics library).
2. Draw a cloud shape (or use a cloud image).
3. Animate the cloud's position over time, moving it across the screen.
4. Add subtle variations to the cloud's shape or size to simulate its changing form. This could involve very slight scaling or warping.
5. Consider adding a slight, slow rotation.
6. Adjust the speed and path of the cloud for a realistic effect.
7. Export the animation.

Source Code: (Example using keyframes and motion tweening - conceptual)

```
// Conceptual code (e.g., Animate timeline)
// Frame 1: Cloud at starting position
// Frame 100: Cloud at ending position (translated)
// Apply motion tween to the cloud instance
// (Optional) Add slight scaling/warping animation
```

Input: (None, the input is the user's design of the cloud and its movement)

Expected Output: An animation of a cloud moving across the screen, possibly changing shape slightly.

6b) Simulations of Change of Shapes

Title: Shape Transformation Simulation

Aim: To simulate changes in shapes.

Procedure:

8. Choose animation software.
9. Create a basic shape (e.g., a square, circle, triangle).
10. Animate the shape transforming into another shape. This could involve:

    Scaling (increasing/decreasing size)

    Rotation

    Morphing (changing the shape's vertices)

11. Use techniques like tweening or keyframe animation to create smooth transitions.
12. Experiment with different shape transformations.

13. Export the animation.

Source Code: (Example using a graphics library - conceptual)

```
// Conceptual code
function animateShapeChange(shape1, shape2, duration) {
  for (time = 0; time < duration; time++) {
     // Calculate intermediate shape
     intermediateShape = interpolate(shape1, shape2, time / duration);
     drawShape(intermediateShape);
  }
}
```

Input: (None, the input is the user's choice of shapes and transformations)

Expected Output: Animations showing one shape transforming into another.

Lab 7: Fan Blade Animation

Title: Fan Blade Animation

Aim: To draw fan blades and animate them with proper rotation.

Procedure:

1. Choose animation software (e.g., Flash/Animate).
2. Draw the shape of a fan blade.
3. Create multiple instances or copies of the blade to form the fan (usually 3 or 4 blades).
4. Position the blades around a central point.
5. Animate the rotation of the fan blades around the central point.
6. Set the rotation speed and direction.
7. Add any desired effects (e.g., motion blur).
8. Export the animation.

Source Code: (Example using rotation animation - conceptual)

```
// Conceptual code (e.g., Animate)
for (angle = 0; angle < 360 * numberOfRotations; angle += rotationSpeed) {
  // Rotate each fan blade by 'angle'
  rotateBlade(blade1, angle);
  rotateBlade(blade2, angle);
  rotateBlade(blade3, angle);
  // (Optional) Apply motion blur
  updateDisplay();
}
```

Input: (None, the input is the user's design of the fan blade and the rotation speed)

Expected Output: An animation of a fan with rotating blades.

Lab 8: Text Animation

Title: Welcome Text Animation

Aim: To create an animation with specific text effects: letters appearing one by one, and changing fill color.

Procedure:

1. Choose animation software.
2. Type the word "Welcome".
3. Separate the letters into individual text objects or layers.
4. Animate the appearance of each letter, one after the other.
5. After the full word is displayed, animate the fill color of the text, changing it to a different color.
6. Control the timing of the letter appearance and color change.
7. Export the animation.

Source Code: (Example - conceptual)

```
// Conceptual code
letters = ["W", "e", "l", "c", "o", "m", "e"];
for (i = 0; i < letters.length; i++) {
  // Show letter[i] after delay
  displayLetter(letters[i], i * letterDelay);
}
waitFor(wordDisplayDuration);
// Change color of all letters
animateTextColor(newColor, colorChangeDuration);
```

Input: The word "Welcome."

Expected Output: An animation where the letters of "Welcome" appear one by one, and then the entire word changes color.

Lab 9: Animated Cursor

Title: Animated Cursor

Aim: To create an animated cursor using startDrag("ss", true); and Mouse.hide(); (This suggests ActionScript, so Flash/Animate).

Procedure:

1. Open Adobe Flash/Animate.
2. Create a movie clip symbol for the custom cursor. Draw the cursor shape within the movie clip.
3. Write ActionScript code:

   Mouse.hide(); to hide the default mouse cursor.

   Use startDrag("ss", true); (assuming "ss" is the instance name of the custom cursor movie clip) to attach the custom cursor to the mouse.

   Potentially, add animation *within* the cursor movie clip (e.g., color changes, shape changes)

4. Test the animation.
5. Export the animation (if needed for web use).

Source Code: (ActionScript - Example)

```
// ActionScript (Flash/Animate)
Mouse.hide(); // Hide the default cursor

// Assuming your custom cursor movie clip's instance name is
"customCursor_mc"
customCursor_mc.startDrag(true); // Attach to mouse, with lock center

// Optional:  If you have animation *inside* the customCursor_mc:
// customCursor_mc.play();  // If the cursor itself is animated
onMouseMove = function() {
   customCursor_mc._x = _xmouse;
   customCursor_mc._y = _ymouse;
}
```

Input: (None, the input is the user's design of the cursor)

Expected Output: The default mouse cursor is hidden, and a custom animated cursor moves with the mouse.

Lab 10: Poster Design for Print and Web

Title: Poster Design for 2024 Election

Aim: To design a poster for the 2024 election and demonstrate the difference in resolution and quality for print and web.

Procedure:

1. Choose a graphics editor (e.g., Photoshop, Illustrator, or GIMP).
2. Design the poster content (text, images, layout) related to the 2024 election.
3. **For Print:**

   Create the poster at a high resolution (e.g., 300 dpi) and in a suitable color mode (CMYK).

   Save the poster in a print-ready format (e.g., PDF, TIFF).

4. **For Web:**

   Create a separate version of the poster (or resize the print version) at a lower resolution (e.g., 72 dpi) and in a web-friendly color mode (RGB).

   Optimize the image for web use (e.g., using JPEG or PNG format).

5. Prepare a document or presentation to show the two versions side-by-side, highlighting the differences in resolution, color, and overall quality.

Source Code: (This is a design task, so the "source code" is the design file in the graphics editor. The procedure outlines the steps.)

Input: Information and assets related to the 2024 election.

Expected Output: Two versions of the election poster (one for print, one for web), and a comparison demonstrating the differences in quality.

Lab 11: 3D Animation from AI-Generated Image

Title: 3D Animation from an AI-Generated Image

Aim: To create a 3D animation using an AI-generated image as a base or inspiration.

Procedure:

1. Use an AI image generation tool (e.g., DALL-E, Midjourney, Stable Diffusion) to create an image. The image should have elements that could be interpreted in 3D.
2. Choose a 3D modeling and animation software (e.g., Blender, Maya, 3ds Max).
3. Import the AI-generated image into the 3D software. This might involve using it as a texture, a reference for modeling, or a background.
4. Model 3D objects based on elements in the AI-generated image.
5. Animate the 3D objects. This could involve:

    Moving objects in 3D space.

    Changing the shape of objects over time.

    Applying materials and textures.

    Setting up lighting and cameras.

6. Render the 3D animation.
7. Export the animation in a suitable format (e.g., video).

Source Code: (This is heavily software-dependent. Blender uses Python scripting, for example. The "source code" includes the 3D model files, animation settings, and any scripts used.)

Input: An AI-generated image.

Expected Output: A 3D animation that is derived from or inspired by the AI-generated image.

Lab 12: Animation for Banning Mobile Phones

Title: Animation for Banning Mobile Phones

Aim: To design an animation that conveys a message about banning mobile phones (presumably in a specific context, like schools or meetings).

Procedure:

1. Choose animation software.
2. Develop a concept or storyboard for the animation. Consider:

    The target audience.

    The message to be conveyed.

    Visual metaphors or symbols.

3. Create the visual elements (characters, objects, backgrounds) for the animation.
4. Animate the elements to tell a story or convey the message. Examples:

    A phone distracting someone.

    A phone being silenced or discarded.

    People interacting positively without phones.

5. Add sound effects or music if desired.
6. Export the animation.

Source Code: (Depends on the software. Could involve keyframes, scripting, etc.)

Input: (None, the input is the user's concept and message)

Expected Output: An animation that promotes the idea of banning mobile phones in a specific context.

Lab 13: Audio and Web Development

13a) Piano Simulation

Title: Piano Key Simulation

Aim: To study the notes of a piano and simulate them using a keyboard, and save the file.

Procedure:

1. Choose a software or programming language that can handle audio playback and keyboard input (e.g., a programming language with an audio library, or a digital audio workstation (DAW)).
2. Obtain or create digital audio samples for each piano note.
3. Map keyboard keys to the corresponding piano notes.
4. Write code or configure the software to play the correct audio sample when a key is pressed.
5. Create a user interface (if necessary) to visualize the piano keyboard.
6. Implement functionality to record and save the played notes (e.g., as a MIDI file or audio recording).

Source Code: (Example using Python and a simple audio library (conceptual))

```
# Conceptual Python code with a hypothetical audio library
import audio
import keyboard

notes = {
    'a': 'C4.wav',
    's': 'D4.wav',
    'd': 'E4.wav',
    # ...
}

def play_note(key):
    if key in notes:
        audio.playSound(notes[key])

def record_notes():
    recording = []
    while True:
        key = keyboard.readKey()
        if key == 'q':  # Example: 'q' to stop recording
            break
        if key in notes:
            play_note(key)
            recording.append((key, time.time())) # Store key and
timestamp
    return recording

# Main program
while True:
    key = keyboard.readKey()
    if key == 'r':
       recorded_data = record_notes()
       #save recorded_data
    elif key in notes:
        play_note(key)
```

Input: Keyboard input.

Expected Output: A program or application that simulates a piano keyboard, plays notes when keys are pressed, and can record and save the played notes.

13b) Institution Web Page

Title: Institution Web Page

Aim: To create a web page for an institution, containing branches of study and links to other web pages.

Procedure:

1. Use HTML to structure the web page content. 2. Use CSS to style the page (layout, colors, fonts). 3. Create sections for each branch of study offered by the institution. 4. For each branch, provide a brief description and links to relevant pages (e.g., department pages, course details). 5. Include at least 10 links to other web pages, which could be: * Internal links to other pages within the institution's website. * External links to relevant resources, organizations, or related websites. 6. Ensure the page is well-organized, easy to navigate, and visually appealing. 7. Validate the HTML and CSS code.

Source Code: (HTML, CSS - Example)

```
<!DOCTYPE html>
<html>
<head>
    <title>Institution Name</title>
    <style>
        /* Basic CSS (Add more detailed styling) */
        body { font-family: sans-serif; }
        nav ul { list-style-type: none; }
        nav ul li { display: inline; margin-right: 10px; }
    </style>
</head>
<body>
    <header>
        <h1>Institution Name</h1>
        <nav>
            <ul>
                <li><a href="#home">Home</a></li>
                <li><a href="#branches">Branches of Study</a></li>
                <li><a href="#about">About Us</a></li>
                <li><a href="https://www.example.com">External Link
1</a></li>
                 <li><a href="https://www.example2.com">External Link
2</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <section id="home">
            <h2>Welcome</h2>
            <p>Welcome to our institution's website.</p>
        </section>

        <section id="branches">
```

```
            <h2>Branches of Study</h2>
            <ul>
                <li>
                    <h3>Computer Science</h3>
                    <p>Description of Computer Science program.</p>
                    <a href="/cs_dept">Computer Science Department</a>
                </li>
                <li>
                    <h3>Engineering</h3>
                    <p>Description of Engineering program.</p>
                    <a href="/engineering_dept">Engineering
Department</a>
                </li>
                 <li>
                    <h3>Business</h3>
                    <p>Description of Business program.</p>
                    <a href="/business_dept">Business Department</a>
                </li>
                 <li>
                    <h3>Medicine</h3>
                    <p>Description of Medicine program.</p>
                    <a href="/medicine_dept">Medicine Department</a>
                </li>
                 <li>
                    <h3>Law</h3>
                    <p>Description of Law program.</p>
                    <a href="/law_dept">Law Department</a>
                </li>
                 <li>
                    <h3>Arts</h3>
                    <p>Description of Arts program.</p>
                    <a href="/arts_dept">Arts Department</a>
                </li>
                 <li>
                    <h3>Science</h3>
                    <p>Description of Science program.</p>
                    <a href="/science_dept">Science Department</a>
                </li>
                 <li>
                    <h3>Education</h3>
                    <p>Description of Education program.</p>
                    <a href="/education_dept">Education Department</a>
                </li>
                 <li>
                    <h3>Agriculture</h3>
                    <p>Description of Agriculture program.</p>
                    <a href="/agriculture_dept">Agriculture
Department</a>
                </li>
                 <li>
                    <h3>Fine Arts</h3>
                    <p>Description of Fine Arts program.</p>
                    <a href="/finearts_dept">Fine Arts Department</a>
                </li>
            </ul>
        </section>
         <section id="about">
            <h2>About Us</h2>
            <p>Information about the institution.</p>
        </section>
    </main>

    <footer>
        <p>&copy; 2024 Institution Name</p>
    </footer>
</body>
</html>
```

Input: Content about the institution and its branches of study.

Expected Output: A multi-page website for the institution, with information about its programs and links to other relevant pages.

Lab 14: Working with Dreamweaver

Title: Web Development with Dreamweaver

Aim: To create a minimum of two programs (web pages) using Dreamweaver.

Procedure:

1. Open Dreamweaver.
2. **Program 1: Basic HTML Page**

   Create a new HTML document.

   Use Dreamweaver's visual editor or code view to structure the page with headings, paragraphs, lists, and images.

   Apply basic styling using CSS (either inline, embedded, or linked).

   Save the HTML file.

3. **Program 2: Interactive Page**

   Create another new HTML document.

   Incorporate interactive elements using HTML forms, JavaScript, or Dreamweaver's built-in behaviors. Examples:

      A contact form.

      A simple image rollover effect.

      A dynamic menu.

   Style the page with CSS.

   Save the HTML file.

4. Document the steps taken for each program, including any specific Dreamweaver features used.

Source Code: (HTML, CSS, and potentially JavaScript generated by Dreamweaver. The procedure describes the creation process.)

Input: Content for the web pages (text, images, etc.).

Expected Output: Two web pages created using Dreamweaver: one with basic HTML structure and styling, and another with interactive elements.

Lab 15: Adding Media to Webpages with Dreamweaver

Title: Adding Media to Webpages

Aim: To add video, audio, and animation to webpages using Dreamweaver.

Procedure:

1. Open Dreamweaver.
2. Create a new HTML document.
3. **Adding Video:**

   Use the HTML <video> tag to embed a video file.

   Use Dreamweaver's interface to insert the video and set attributes (e.g., src, controls, width, height).

   Ensure the video file is in a supported format (e.g., MP4).

4. **Adding Audio:**

   Use the HTML <audio> tag to embed an audio file.

   Use Dreamweaver to insert the audio and set attributes (e.g., src, controls).

   Ensure the audio file is in a supported format (e.g., MP3).

5. **Adding Animation:**

   There are several ways to add animation:

   GIF Animation: Insert a pre-made GIF animation using the <img> tag.

   HTML5 Canvas: Use Dreamweaver's code view to add a <canvas> element and write JavaScript to create animations.

   Embedded Animation: Embed animation created with other software (e.g., a Flash animation, if supported, or a Lottie animation) using the appropriate HTML tags.

   Use Dreamweaver to help with the insertion and any necessary code.

6. Style the page with CSS to integrate the media elements.
7. Save the HTML file.

Source Code: (HTML, CSS, and potentially JavaScript generated by Dreamweaver. The procedure outlines the process.)

Input: Video, audio, and animation files.

Expected Output: A webpage that includes embedded video, audio, and animation.