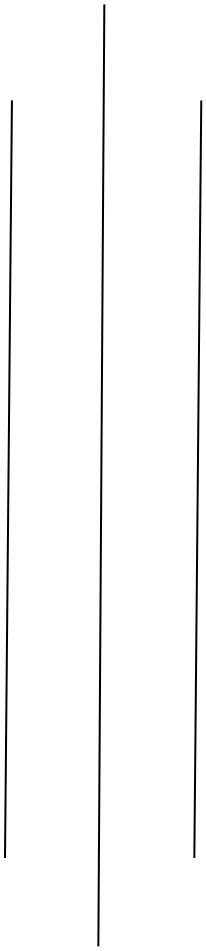
# Nagarjuna College of Information Technology

***(Affiliated to Tribhuvan University)***

***Shankhamul, Lalitpur***

B.Sc.CSIT Semester-V



*Lab Reports of Multimedia Computing*

| **Submitted By:** | **Submitted to:** |
|---|---|
| Manish Shiwakoti<br>*(Student of B.Sc.CSIT Semester-V)*<br>**Roll No:** 26448/077<br>**Subject Code:** CSC. 319<br>**T.U. Registration No.:** 5-2-429-11-2020 | Sudarshan Sharma<br>*(Lecturer of Multimedia Computing)* |

# **Lab 1:** Conversion of Original Image to Black and White Image Using Python

**Source Code:**

```
import cv2
import os
import numpy as np

# here cv2, os and numpy are libraries of python
# Here, img is an object
# img=cv2.imread('./img/image.jpg')
img=cv2.imread('image.jpg')
# imread is a keyword under cv2 library which reads input as image

#initialImage is the name of transformation of windows
cv2.imshow('initialImage',img)
#img=cv2.resize(img,(400, 500))
# imshow is a keyword under cv2 library which shows/displays greyscaled (Black/White) image as output

image=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
(thresh, bimage1)=cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)
(thresh, bimage2)=cv2.threshold(image, 150, 200, cv2.THRESH_BINARY)

#both_image=np.hstack([image, b image1, bimage2])

cv2.imshow('Black&White_image_with_first_threshold_value', bimage1)
cv2.waitKey(0)

cv2.imshow('Black&White_image_with_second_threshold_value', bimage2)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

**Output:**


**Original Image**


**Black and White First Threshold Image**


**Black and White Second Threshold Image**

# **Lab 2:** Conversion of Original Image to Grayscale Image Using Python

**Source Code:**

```
import cv2
import numpy as np

img=cv2.imread('image.jpg')
cv2.imshow('Original', img)
#image=cv2.resize(img,(500, 700))

# Creating a copy of the Original Image
img2=img.copy()

gray_image=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Conversion of Image into 3 channel for Concatenation Process
img2[:, :, 0]=gray_image
img2[:, :, 1]=gray_image
img2[:, :, 2]=gray_image


cv2.imshow('Grayimage', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

**Output:**


**Original Image**


**Grayscale Image**

# **Lab 3:** Implementation of Huffmann's Coding Using Python

**Source Code:**

```
# A Huffman Tree Node
import heapq

class node:
    def __init__(self, freq, symbol, left=None, right=None):
        # frequency of symbol
        self.freq = freq

        # symbol name (character)
        self.symbol = symbol

        # node left of current node
        self.left = left

        # node right of current node
        self.right = right

        # tree direction (0/1)
        self.huff = ''

    def __lt__(self, nxt):
        return self.freq < nxt.freq

# utility function to print Huffman codes for all symbols in the newly created Huffman tree
def printNodes(node, val=''):
    # huffman code for current node
    newVal = val + str(node.huff)

    # if node is not an edge node, then traverse inside it
    if(node.left):
        printNodes(node.left, newVal)
    if(node.right):
        printNodes(node.right, newVal)

        # if node is edge node, then display its huffman code
    if(not node.left and not node.right):
        print(f"{node.symbol} -> {newVal}")

# characters for huffman tree
chars = ['a', 'b', 'c', 'd', 'e', 'f']

# frequency of characters
freq = [5, 9, 12, 13, 16, 45]

# list containing unused nodes
nodes = []
```

```
# converting characters and frequencies into huffman tree nodes
for x in range(len(chars)):
    heapq.heappush(nodes, node(freq[x], chars[x]))

while len(nodes) > 1:

    # sort all the nodes in ascending order based on their frequency
    left = heapq.heappop(nodes)
    right = heapq.heappop(nodes)

    # assign directional value to these nodes
    left.huff = 0
    right.huff = 1

    # combine the 2 smallest nodes to create
    # new node as their parent
    newNode = node(left.freq+right.freq, left.symbol+right.symbol, left, right)

    heapq.heappush(nodes, newNode)

# Huffman Tree is ready!
printNodes(nodes[0])
```

**Output:**

# **Lab 4:** Implementation of Run Length Encoding (RLE) Using Python

**Source Code:**

```
#Run Length Coding

def encode(message):
    encoded_message = ""
    i = 0

    while (i <= len(message)-1):
        count = 1
        ch = message[i]
        j = i
        while (j < len(message)-1):
            if (message[j] == message[j+1]):
                count = count+1
                j = j+1
            else:
                break
        encoded_message=encoded_message+str(count)+ch
        i = j+1
    return encoded_message

#Provide different values for message and test your program
encoded_message=encode("ABBBBCCCCCCCCAB")
print(encoded_message)
```

**Output:**

# **Lab 5:** Creating UX Design Using Figma

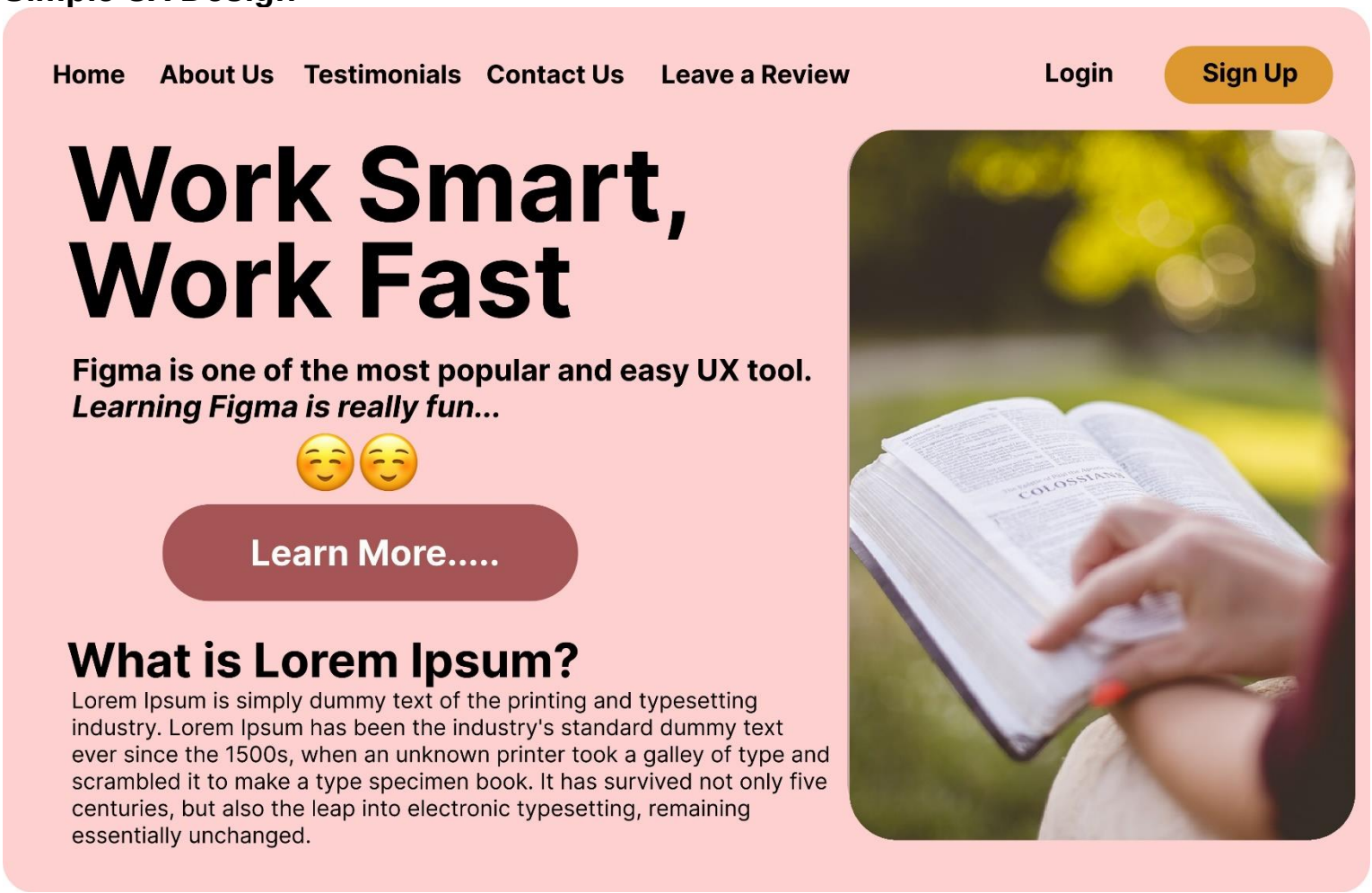
**Introduction:**

Figma is a cloud-based design and prototyping tool used for user interface (UI) and user experience (UX) design. It is particularly popular among designers and teams for its collaborative features, real-time editing capabilities, and cross-platform accessibility.

## Steps of Creating UX Design

1. Initially, get logged in into the Figma through Gmail account by tapping into **Get Started for free.**
2. After you log in to Figma, click on Design File to create a New Design.
3. Select on Frame to create new Frame.
4. Select Device Frame Design to Create Frame related to various devices like Phone, Tablet, Desktop, Presentation, Watch, etc.
5. Finally, start design inside the frames by adding various shapes, colors, texts and so on.

## Simple UX Design



# **Lab 6:** Reducing Noise/Audio Using Audacity

**Introduction:**

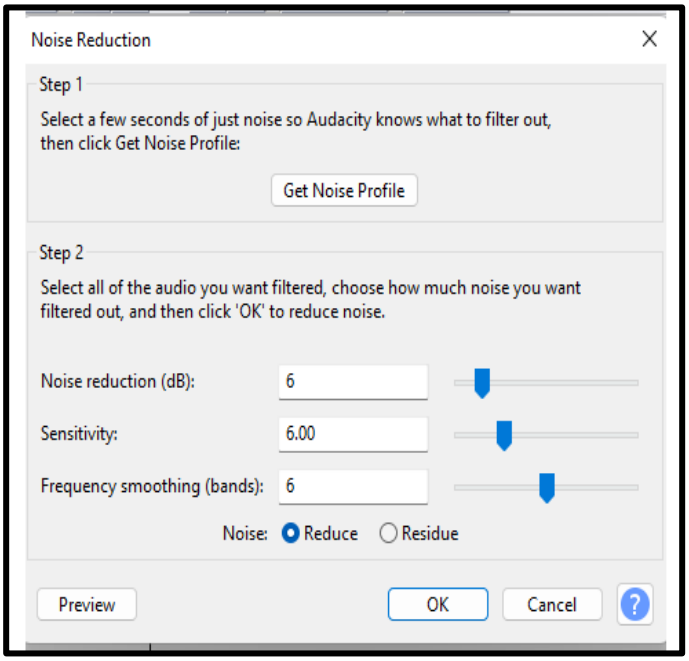
Audacity is a free, open-source digital audio editing and recording software. It is widely used for a variety of audio editing tasks, such as recording and editing podcasts, creating music, digitizing analog recordings, and more. Audacity supports multiple tracks and offers a range of features for manipulating audio, including cutting, copying, and pasting sections of audio, applying various effects, adjusting volume levels, and more.

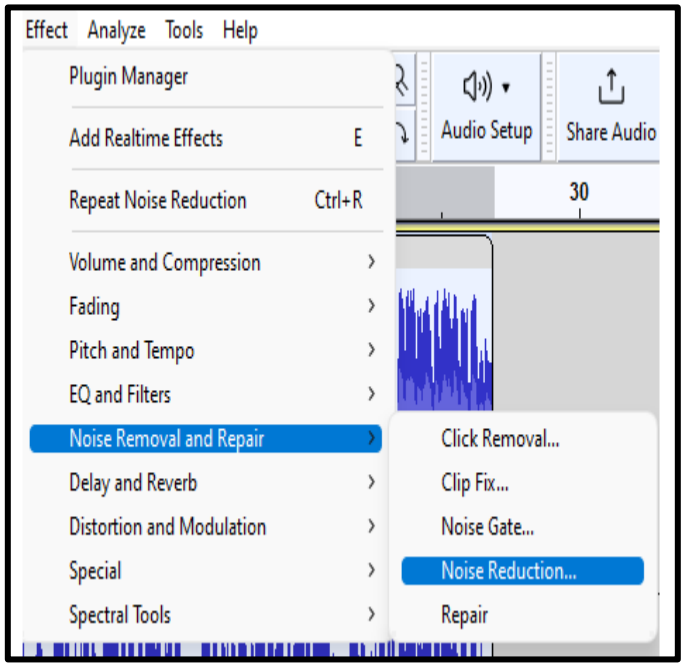
## Steps for Noise Reduction

Noise reduction is a delicate process, and overdoing it can result in a loss of audio quality. It's essential to strike a balance between reducing noise and preserving the clarity of the desired audio. Additionally, always keep a backup of your original file before making significant edits, so you can revert to it if needed.

Following are the steps for applying Noise Reduction:

1. **Open Your Audio File:**
   - Launch Audacity and open the audio file you want to edit.
2. **Select a Section with Only Noise:**
   - Find a section in the recording where there is only background noise **(***no desired audio***)**. This will be used to create a noise profile.
3. **Capture Noise Profile:**
   - Highlight the section with only noise.
   - Go to "*Effect*" in the menu bar and select "*Noise Reduction*."
   - Click on the "*Get Noise Profile*" button.
4. **Apply Noise Reduction:**
   - After capturing the noise profile, select the entire audio or the specific parts where you want to reduce noise.
   - Again, go to ***Effect*>Noise Removal and Repair>*Noise Reduction*** and select the noise part of the audio as noise profile.




   - Adjust the settings as needed. Generally, the default settings work well for many cases.
   - Click "***OK***" to apply the noise reduction.

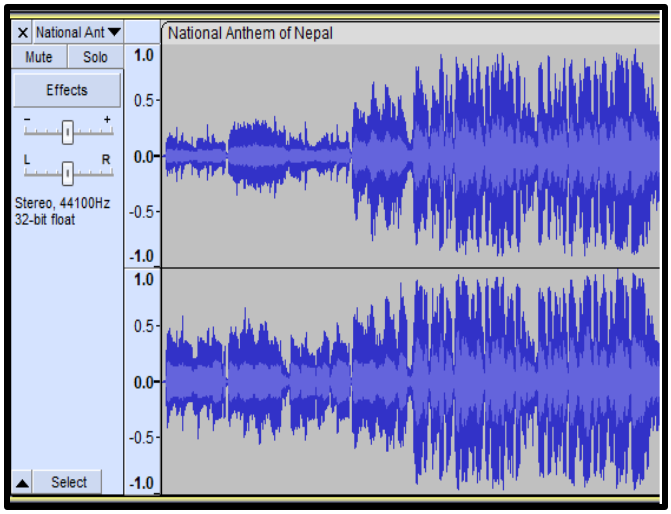



**Image Before Reduction**

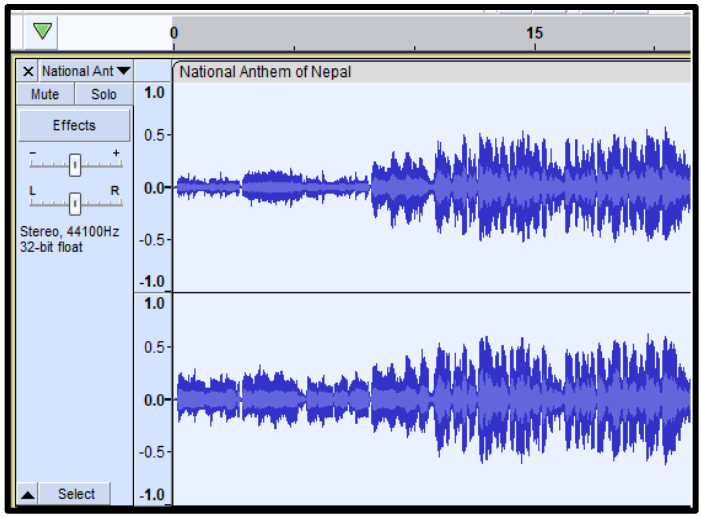



**Image After Reduction**

5. **Listen and Adjust:**
   - Listen to the edited audio to make sure the noise reduction is satisfactory. If necessary, you can undo the effect and adjust the settings to achieve the desired result.
6. **Export the Edited File:**
   - Once you are satisfied with the noise reduction, export the edited audio file. Go to **File>*Export*** and choose the desired file format.

# **Lab 7:** Mixing Two Sounds Using Audacity

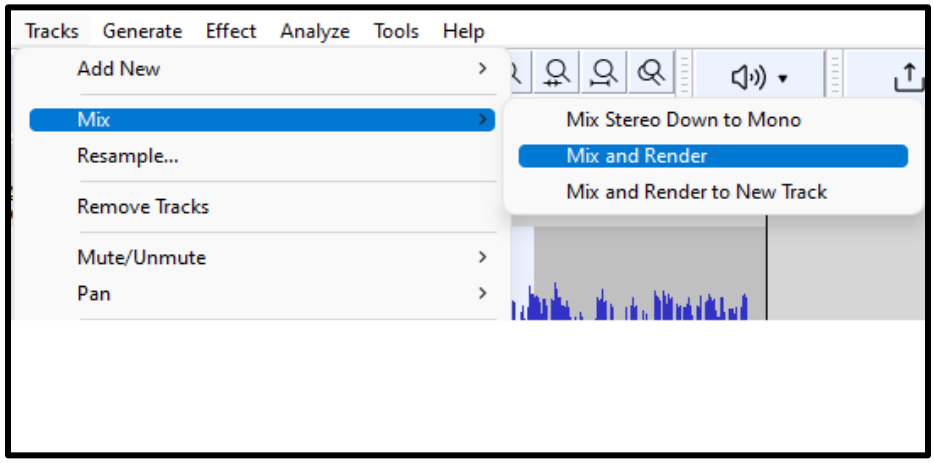
**Introduction:**

Audacity is a free, open-source digital audio editing and recording software. It is widely used for a variety of audio editing tasks, such as recording and editing podcasts, creating music, digitizing analog recordings, and more. Audacity supports multiple tracks and offers a range of features for manipulating audio, including cutting, copying, and pasting sections of audio, applying various effects, adjusting volume levels, and more.

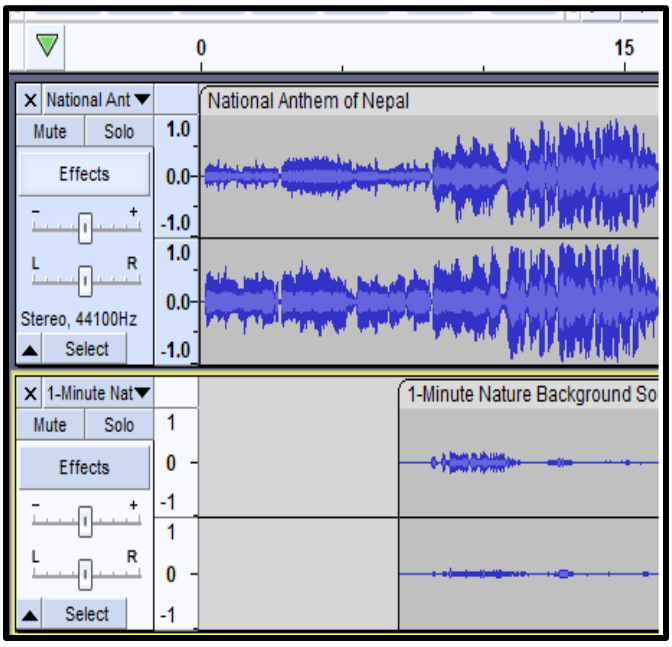
## Steps of Mixing Up Sound

Noise reduction is a delicate process, and overdoing it can result in a loss of audio quality. It's essential to strike a balance between reducing noise and preserving the clarity of the desired audio. Additionally, always keep a backup of your original file before making significant edits, so you can revert to it if needed.

1. Select multiple sound tracks to merge.
2. Click on **Tracks>Mix>Mix and Render.**




3. Here, is the before and after of sound mixing.




Before Mixing


After Mixing

# **Lab 8:** Creating Simple Animation Using Adobe Animate

## Introduction:

Adobe Animate is a multimedia authoring and animation software developed by Adobe Inc. It is used for creating interactive content, animations, and multimedia applications. Originally known as Adobe Flash Professional, Adobe rebranded it as Adobe Animate to reflect its evolving capabilities beyond just Flash.

Adobe Animate allows users to design and develop a wide range of content, including vector graphics, animations, interactive games, and web applications. It supports various output formats, such as SWF (Shockwave Flash), HTML5, WebGL, and more. This flexibility enables creators to deliver content across different platforms and devices, including web browsers, desktop applications, and mobile devices.

The software provides a timeline-based interface for animating objects, along with support for scripting languages like ActionScript (for Flash-based projects) and JavaScript (for HTML5-based projects). Adobe Animate also integrates with other Adobe Creative Cloud applications, allowing for seamless workflows with software like Adobe Illustrator and Adobe Photoshop.

While, Adobe Animate is widely used for creating web-based animations and interactive content, it's worth noting that the use of Adobe Flash technology has been declining in recent years due to changes in web standards, security concerns, and the rise of alternative technologies like HTML5. As a result, Adobe Animate has adapted to support modern web standards and remains a valuable tool for multimedia and animation design.

## Key Features of Adobe Animate

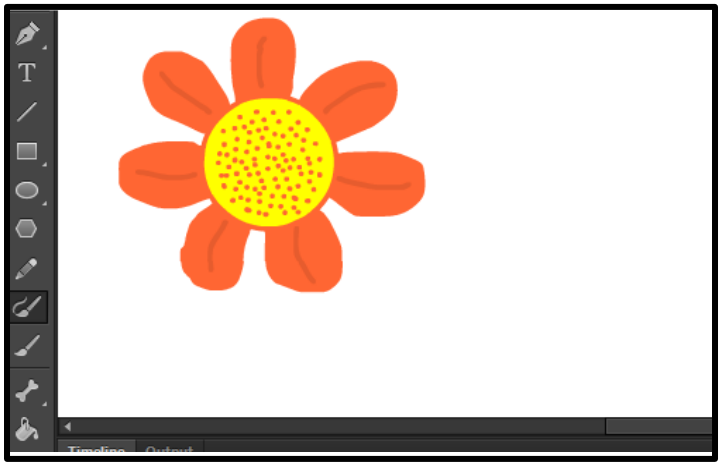
### 1. Vector Graphics

It supports vector graphics, allowing users to create scalable and high-quality visuals. The user drawing is converted into vector graphics automatically.

**Step I:** Select the brush tool from the sidebar or press 'B'.

**Step II:** Draw the desired shape on the canvas.

**Step III:** Fill the shape with desired color from fill properties.



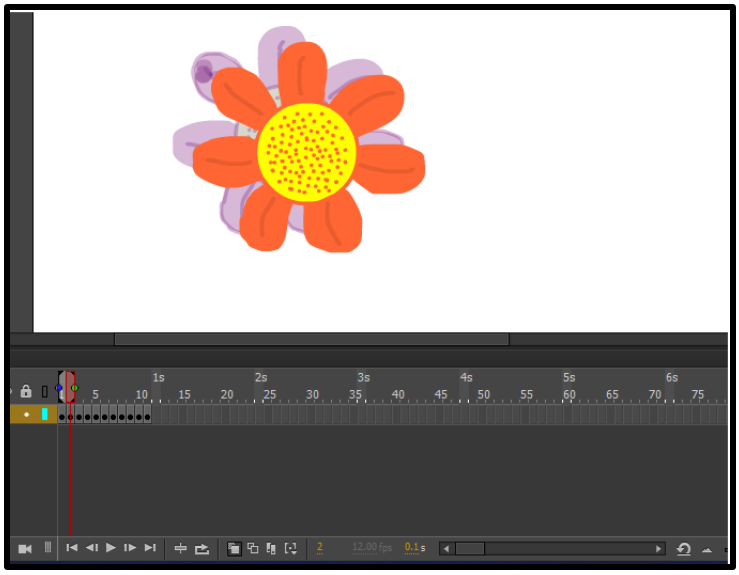
### 2. Animation

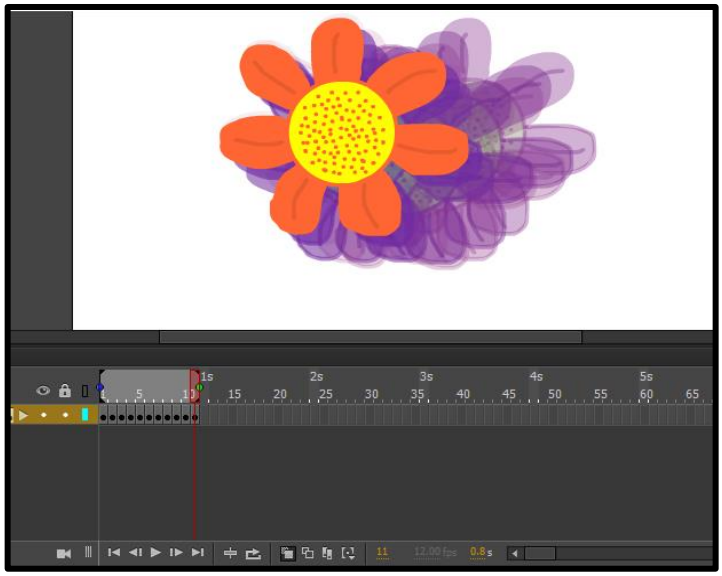
It has powerful timeline-based animation capabilities for creating dynamic and interactive content. Let's animate a swinging flower.

**Step I:** Draw an object on the current frame.

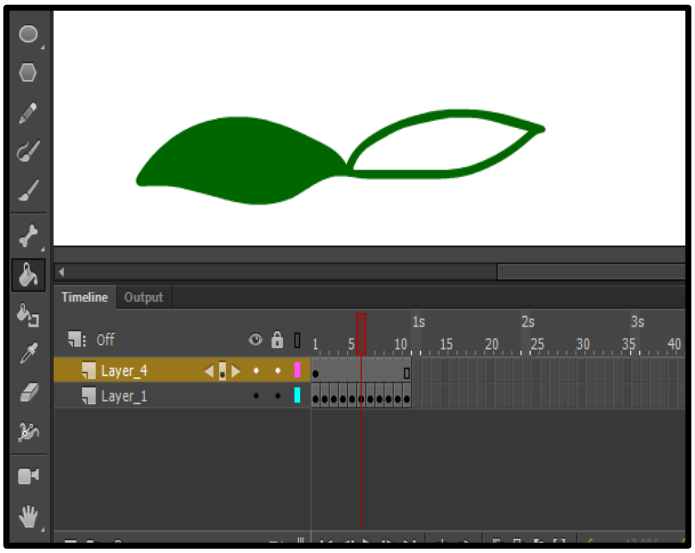
**Step II:** Press **'F7'** to create the next frame. Press **','** and **'.'** keys to step through the keyframes.

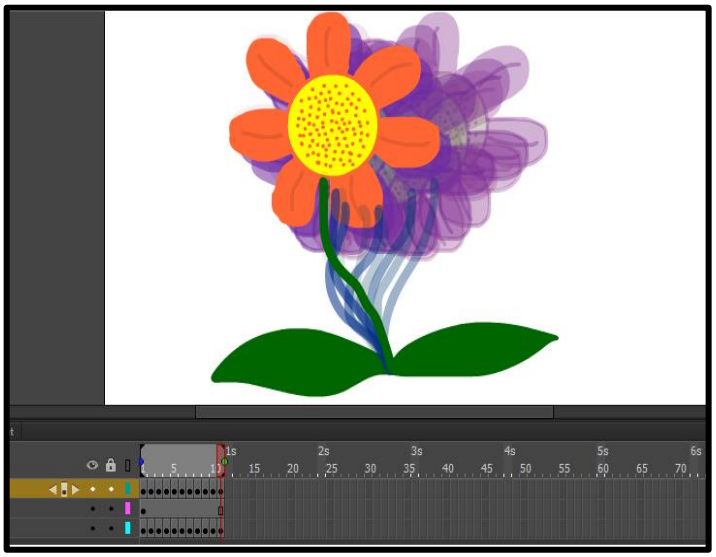
**Step III:** Enable onion skin to see the shadow of the previous frame and draw for all different frames.





**Step IV:** Create a new layer and extend the keyframe for static objects like leafs. Again, in a new layer animate the flower stem frame by frame.

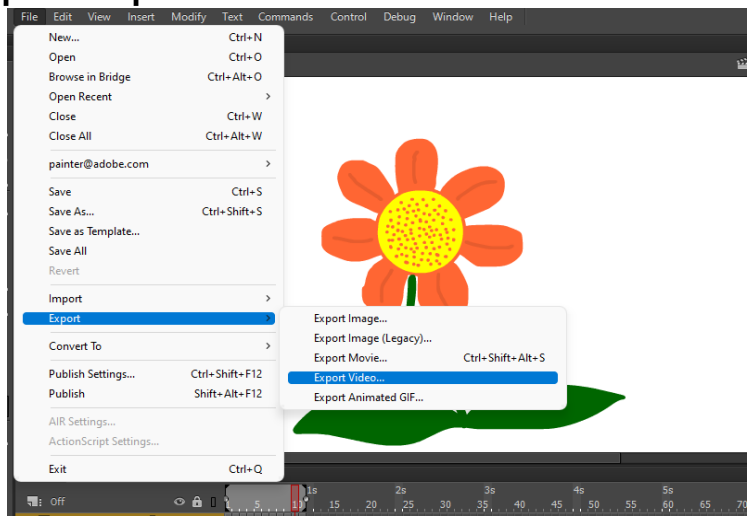


## 3. Export Options

It provides versatile export options, including HTML5 Canvas, WebGL, and video formats, for different platforms.

**Step I:** Click on **File>Export>Export Video.**




**Step II:** Make sure to install and open **Adobe Media Encoder,** then click **Export.**