Linear Algebra Project

Group 9

Group Members

- Malik Saad Ahmed 231224
- Quratulain 231186
- Radeel Ayesha 231176
- Sehar Tariq 231196
- Hadia Ahsheen 231164

Cartoon

The cartoon that we will plot is Olaf, a snowman from frozen. Our reference image is

In [ ]:
```
import matplotlib.pyplot as plt

image = plt.imread('3o9q7j2of9y81 (1).png')

plt.imshow(image)
plt.axis('off')
plt.show()
```

Project Pipeline

Our project is divided into 4 parts

1. Extract coordinate points from a reference image
2.
3. Plot the cartoon using coordinate points manually extracted from the image using matplotlib
4. Apply Expansion by a factor of 4 along y-axis
5. Apply Shear by a factor of 4 along x-axis

Loading The Libraries

The libraries we use are

- Matplotlib
- Numpy
- Scipy

We will import these libraries in python so we can use them to plot our target image

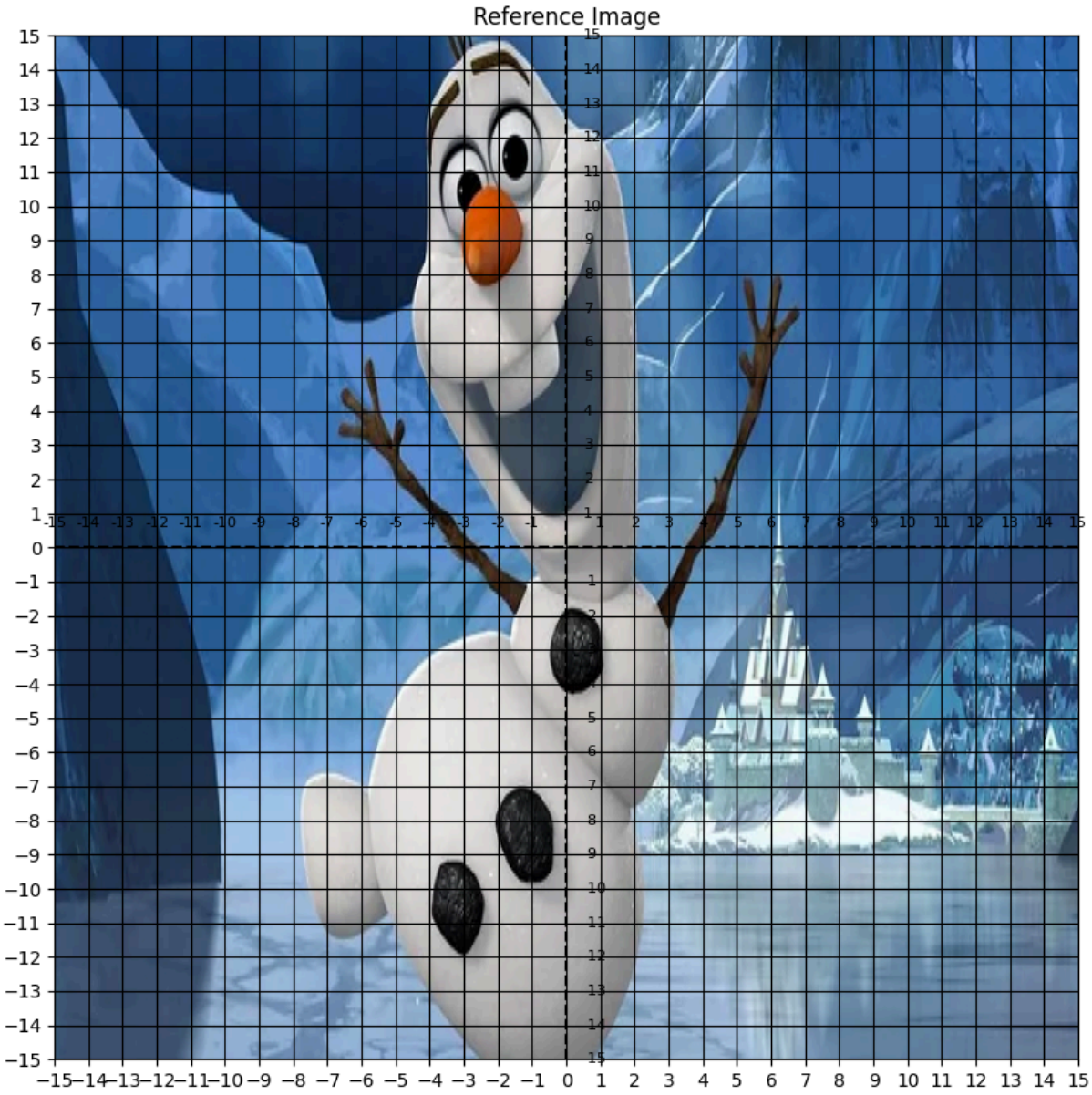
In [1]:
```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib.image as mpimg
import numpy as np
from scipy.interpolate import interp1d
```

Extracting Coordinate Points

We begin by extracting coordinate points from the image. The strategy we followed is that we downloaded an image of OLAF and set it as a background image of an empty matplotlib graph and used plt.gca() method to plot a 15x15 grid on the image. This way we were able to identify the coordinate points efficiently.

In [ ]:
```
bg_img = mpimg.imread('3o9q7j2of9y81 (1).png')
fig, ax = plt.subplots(figsize=(10, 10))
ax.imshow(bg_img, extent=[-15, 15, -15, 15], zorder=0)
plt.axis('equal')
plt.title('Reference Image')
plt.axhline(0, color='black', linestyle='--')
plt.axvline(0, color='black', linestyle='--')
plt.xticks(range(-15, 16, 1))
plt.yticks(range(-15, 16, 1))
plt.xlim(-15, 15)
plt.ylim(-15, 15)
for i in range(-15, 16):
    if i != 0:

        ax.text(i, 0.5, str(i), ha='center', va='bottom', color='black', for
```

```
        ax.text(0.5, i, str(i), ha='left', va='center', color='black', fonts
plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True, color='black')
```



Plotting The Olaf Using X and Y coordinates

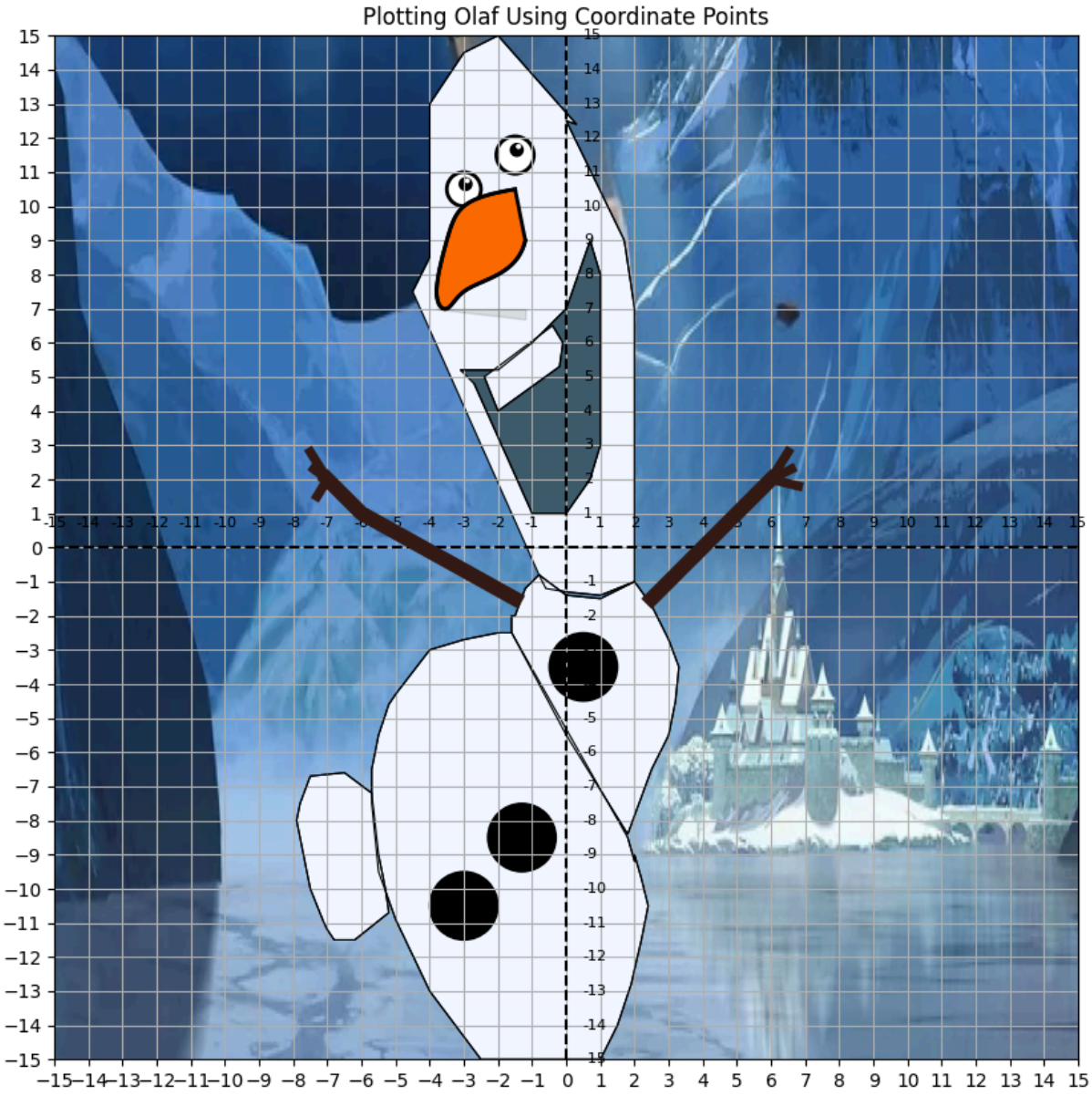
Now, that we have a graph containing the coordinate points, we will manually extract the points and make two lists, one for x coordinates and the other for y cooridnates. This will be manual annotations.

In [7]:
```
bg_img = mpimg.imread('olaf bg.png')
fig, ax = plt.subplots(figsize=(10, 10))

ax.imshow(bg_img, extent=[-15, 15, -15, 15], zorder=0)

# head
x_coord = [-0.8, -0.6,   0,    1,    2,    2,    2,  1.7, 0,     0.3,   -2,
```

```
y_coord = [-0.8, -1.2, -1.3, -1.4,   -1,    0,    7,    9, 12.5,   12.4,   1

plt.fill(x_coord, y_coord, color='#f0f8ff', edgecolor='black', linewidth=0.8
plt.plot(x_coord, y_coord, color='black', linewidth=0.8)

plt.axis('equal')
plt.axhline(0, color='black', linestyle='--')
plt.axvline(0, color='black', linestyle='--')
plt.xticks(range(-15, 16, 1))
plt.yticks(range(-15, 16, 1))
plt.xlim(-15, 15)
plt.ylim(-15, 15)
plt.grid(True)


for i in range(-15, 16):
    if i != 0:

        ax.text(i, 0.5, str(i), ha='center', va='bottom', color='black', for

        ax.text(0.5, i, str(i), ha='left', va='center', color='black', fonts
plt.gca().set_aspect('equal', adjustable='box')


# eyes

eye1 = patches.Circle((-1.5, 11.5), radius=0.55, edgecolor='black', facecolo
eye2 = patches.Circle((-3.0, 10.5), radius=0.5, edgecolor='black', facecolor
ax.add_patch(eye1)
ax.add_patch(eye2)

pupil1 = patches.Circle((-1.45, 11.65), radius=0.18, edgecolor='black', face
pupil2 = patches.Circle((-2.95, 10.65), radius=0.18, edgecolor='black', face
ax.add_patch(pupil1)
ax.add_patch(pupil2)

#eye highlights
highlight1 = patches.Circle((-1.4, 11.75), radius=0.05, color='white')
highlight2 = patches.Circle((-2.9, 10.75), radius=0.05, color='white')
ax.add_patch(highlight1)
ax.add_patch(highlight2)
#nose

x_nose = [-1.5, -3.0, -3.5, -3.8, -3.5, -3.0, -2.0, -1.2]
y_nose = [10.5, 10.0, 9.0, 7.5, 7.0, 7.5, 8.0, 9.0]

t = np.linspace(0, 1, len(x_nose))
t_smooth = np.linspace(0, 1, 200)
cs_x = interp1d(t, x_nose, kind='cubic')
cs_y = interp1d(t, y_nose, kind='cubic')
x_smooth = cs_x(t_smooth)
y_smooth = cs_y(t_smooth)

plt.fill(x_smooth, y_smooth, color='#FD6A02', edgecolor='black', linewidth=2
plt.plot(x_smooth, y_smooth, color='black', linewidth=2, zorder=4)
```

```
shadow_x = np.linspace(min(x_smooth), max(x_smooth), 50)
shadow_y = np.linspace(min(y_smooth), min(y_smooth) - 0.3, 50)
plt.fill_between(shadow_x, shadow_y, min(y_smooth), color='black', alpha=0.1


#smile

x_smile = [-2.7, -1,0,0.7,1,1,0.7,0,-1,-2, -3.1    ]
y_smile = [ 4.8,  1,1,2  ,3,8,9  ,7,6 ,5.2,5.2]


plt.fill(x_smile, y_smile, color='#3e5a69', edgecolor='black', linewidth=0.8
plt.plot(x_smile, y_smile, color='black', linewidth=0.8)

#teeth
x_teeth = [-2.4 ,-2, -0.2,-0.1,-0.4]
y_teeth = [5    ,4, 5.3 , 6,6.5]


plt.fill(x_teeth, y_teeth, color='#f0f8ff', edgecolor='black', linewidth=0.8
plt.plot(x_teeth, y_teeth, color='black', linewidth=0.8)



#chest
x_chest = [-0.8,-1.2,-1.5,-1.6,-1.6,0 , 1.8  ,2.5 ,3,3.2    , 3.3,3    ,2.5,
y_chest = [-0.8,-1.2,-2  ,-2  ,-2.5,-5.5,-8.4,-6.5,-5.5,-4.5,-3.5,-2.7,-1.7,

plt.fill(x_chest, y_chest, color='#f0f8ff', edgecolor='black', linewidth=0.8
plt.plot(x_chest, y_chest, color='black', linewidth=0.8)


#body
x_body = [-1.6,-2  ,-3  ,-4,-5  ,-5.2,-5.5,-5.7,-5.7,-5.5,-5.0,-4,-2.5,1,1.5
y_body = [-2.5,-2.5,-2.7,-3,-4.3,-4.6,-5.5,-6.5,-7.5,-9.5,-10.9,-13,-15,-15,

plt.fill(x_body, y_body, color='#f0f8ff', edgecolor='black', linewidth=0.8)
plt.plot(x_body, y_body, color='black', linewidth=0.8)

#feet

x_foot = [-5.7, -6.5, -7.5, -7.8, -7.9, -7.7,-7.5,-7.1,-7   ,-6.8 ,-6.2 ,-5.
y_foot = [-7.2, -6.6, -6.7, -7.5, -8  , -9  ,-10 ,-11 ,-11.2,-11.5,-11.5,-10
plt.fill(x_foot, y_foot, color='#f0f8ff', edgecolor='black', linewidth=0.8)
plt.plot(x_foot, y_foot, color='black', linewidth=0.8)

#buttons
button1= patches.Circle((0.5, -3.5), radius=1, edgecolor='black', facecolor=
button2 = patches.Circle((-1.3, -8.5), radius=1, edgecolor='black', facecolo
button3 = patches.Circle((-3, -10.5), radius=1, edgecolor='black', facecolor
ax.add_patch(button1)
ax.add_patch(button2)
ax.add_patch(button3)

# Left arm
x_left_arm = [2.5, 5, 6]
```

```
y_left_arm = [-1.5, 1, 2]
ax.plot(x_left_arm, y_left_arm, color='#341d17', linewidth=8, zorder=5)

ax.plot([6, 6.5], [2, 2.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([6, 6.8], [2, 1.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([6, 6.6], [2, 2.3], color='#341d17', linewidth=5, zorder=5)

#right arm
x_right_arm = [-1.5, -6, -7]
y_right_arm = [-1.5, 1, 2]
ax.plot(x_right_arm, y_right_arm, color='#341d17', linewidth=8, zorder=5)

ax.plot([-7, -7.5], [2, 2.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([-7, -7.3], [2, 1.5], color='#341d17', linewidth=5, zorder=5)
ax.plot([-7, -7.4], [2, 2.3], color='#341d17', linewidth=5, zorder=5)

plt.title('Plotting Olaf Using Coordinate Points')

plt.grid(True)
plt.savefig('final.png')
plt.show()
```



Expanding Along Y Axis

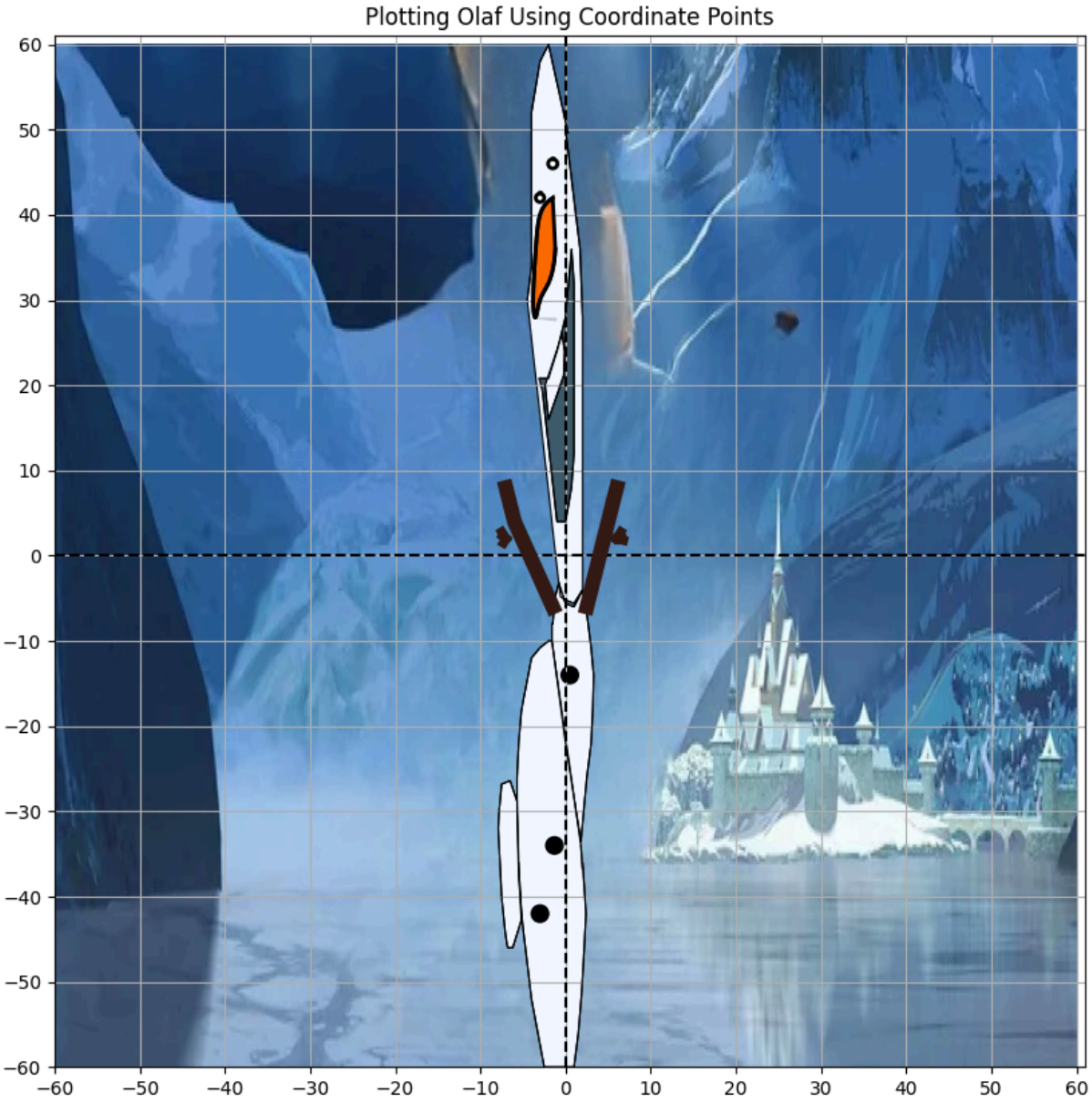
Our next task is expanding the image along y axis by a factor of 4. Simply we have to multiply each y-coordinate by 4. and we will increase the range of grid from 15-15 to 60-60. The reason is that in our original grid the highest value is 15. Multiplying it by 4 yields 60. That is our new grid length

In [ ]:
```
bg_img = mpimg.imread('olaf bg.png')
fig, ax = plt.subplots(figsize=(10, 10))

ax.imshow(bg_img, extent=[-60, 60, -60, 60], zorder=0)

# head
x_coord = [-0.8, -0.6,   0,    1,    2,    2,    2,  1.7, 0,     0.3,   -2,
y_coord = [x*4 for x in [-0.8, -1.2, -1.3, -1.4,   -1,    0,    7,    9, 12.

plt.fill(x_coord, y_coord, color='#f0f8ff', edgecolor='black', linewidth=0.8
```

```
plt.plot(x_coord, y_coord, color='black', linewidth=0.8)

plt.axis('equal')
plt.axhline(0, color='black', linestyle='--')
plt.axvline(0, color='black', linestyle='--')
plt.xticks(range(-60, 61, 10))
plt.yticks(range(-60, 61, 10))
plt.xlim(-60, 61)
plt.ylim(-60, 61)
plt.grid(True)


plt.gca().set_aspect('equal', adjustable='box')


# eyes

eye1 = patches.Circle((-1.5, 11.5*4), radius=0.55, edgecolor='black', facecc
eye2 = patches.Circle((-3.0, 10.5*4), radius=0.5, edgecolor='black', facecol
ax.add_patch(eye1)
ax.add_patch(eye2)

pupil1 = patches.Circle((-1.45, 11.65*4), radius=0.18, edgecolor='black', fa
pupil2 = patches.Circle((-2.95, 10.65*4), radius=0.18, edgecolor='black', fa
ax.add_patch(pupil1)
ax.add_patch(pupil2)

#eye highlights
highlight1 = patches.Circle((-1.4, 11.75*4), radius=0.05, color='white')
highlight2 = patches.Circle((-2.9, 10.75*4), radius=0.05, color='white')
ax.add_patch(highlight1)
ax.add_patch(highlight2)
#nose

x_nose = [-1.5, -3.0, -3.5, -3.8, -3.5, -3.0, -2.0, -1.2]
y_nose = [4*x for x in [10.5, 10.0, 9.0, 7.5, 7.0, 7.5, 8.0, 9.0]]

t = np.linspace(0, 1, len(x_nose))
t_smooth = np.linspace(0, 1, 200)
cs_x = interp1d(t, x_nose, kind='cubic')
cs_y = interp1d(t, y_nose, kind='cubic')
x_smooth = cs_x(t_smooth)
y_smooth = cs_y(t_smooth)

plt.fill(x_smooth, y_smooth, color='#FD6A02', edgecolor='black', linewidth=2
plt.plot(x_smooth, y_smooth, color='black', linewidth=2, zorder=4)

shadow_x = np.linspace(min(x_smooth), max(x_smooth), 50)
shadow_y = np.linspace(min(y_smooth), min(y_smooth) - 0.3, 50)
plt.fill_between(shadow_x, shadow_y, min(y_smooth), color='black', alpha=0.1


#smile

x_smile = [-2.7, -1,0,0.7,1,1,0.7,0,-1,-2, -3.1    ]
y_smile = [x*4 for x in [ 4.8,  1,1,2  ,3,8,9  ,7,6 ,5.2,5.2]]
```

```
plt.fill(x_smile, y_smile, color='#3e5a69', edgecolor='black', linewidth=0.8
plt.plot(x_smile, y_smile, color='black', linewidth=0.8)

#teeth
x_teeth = [-2.4 ,-2, -0.2,-0.1,-0.4]
y_teeth =[x*4 for x in  [5    ,4, 5.3 , 6,6.5]]


plt.fill(x_teeth, y_teeth, color='#f0f8ff', edgecolor='black', linewidth=0.8
plt.plot(x_teeth, y_teeth, color='black', linewidth=0.8)



#chest
x_chest = [-0.8,-1.2,-1.5,-1.6,-1.6,0 , 1.8  ,2.5 ,3,3.2    , 3.3,3    ,2.5,
y_chest = [x*4 for x in [-0.8,-1.2,-2  ,-2  ,-2.5,-5.5,-8.4,-6.5,-5.5,-4.5,-

plt.fill(x_chest, y_chest, color='#f0f8ff', edgecolor='black', linewidth=0.8
plt.plot(x_chest, y_chest, color='black', linewidth=0.8)


#body
x_body = [-1.6,-2  ,-3  ,-4,-5  ,-5.2,-5.5,-5.7,-5.7,-5.5,-5.0,-4,-2.5,1,1.5
y_body =[x*4 for x in  [-2.5,-2.5,-2.7,-3,-4.3,-4.6,-5.5,-6.5,-7.5,-9.5,-10.

plt.fill(x_body, y_body, color='#f0f8ff', edgecolor='black', linewidth=0.8)
plt.plot(x_body, y_body, color='black', linewidth=0.8)

#feet

x_foot = [-5.7, -6.5, -7.5, -7.8, -7.9, -7.7,-7.5,-7.1,-7   ,-6.8 ,-6.2 ,-5.
y_foot =[x*4 for x in  [-7.2, -6.6, -6.7, -7.5, -8  , -9  ,-10 ,-11 ,-11.2,-
plt.fill(x_foot, y_foot, color='#f0f8ff', edgecolor='black', linewidth=0.8)
plt.plot(x_foot, y_foot, color='black', linewidth=0.8)

#buttons
button1= patches.Circle((0.5, -3.5*4), radius=1, edgecolor='black', facecolo
button2 = patches.Circle((-1.3, -8.5*4), radius=1, edgecolor='black', facecc
button3 = patches.Circle((-3, -10.5*4), radius=1, edgecolor='black', facecol
ax.add_patch(button1)
ax.add_patch(button2)
ax.add_patch(button3)

# Left arm
x_left_arm = [2.5, 5, 6]
y_left_arm = [4*x for x in [-1.5, 1, 2]]
ax.plot(x_left_arm, y_left_arm, color='#341d17', linewidth=8, zorder=5)

ax.plot([6, 6.5], [2, 2.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([6, 6.8], [2, 1.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([6, 6.6], [2, 2.3], color='#341d17', linewidth=5, zorder=5)

#right arm
x_right_arm = [-1.5, -6, -7]
```

```
y_right_arm = [4*x for x in [-1.5, 1, 2]]
ax.plot(x_right_arm, y_right_arm, color='#341d17', linewidth=8, zorder=5)

ax.plot([-7, -7.5], [2, 2.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([-7, -7.3], [2, 1.5], color='#341d17', linewidth=5, zorder=5)
ax.plot([-7, -7.4], [2, 2.3], color='#341d17', linewidth=5, zorder=5)

plt.title('Plotting Olaf Using Coordinate Points')

plt.grid(True)
plt.savefig('final.png')
plt.show()
```



Applying Shear along x-axis

Now we have to apply shear by a factor of 4 along x-axis. The formula to apply shear is

T(X,Y) = (x+yk, y)

Shear is a type of linear transformation that shifts points along a line while keeping other points on the line fixed

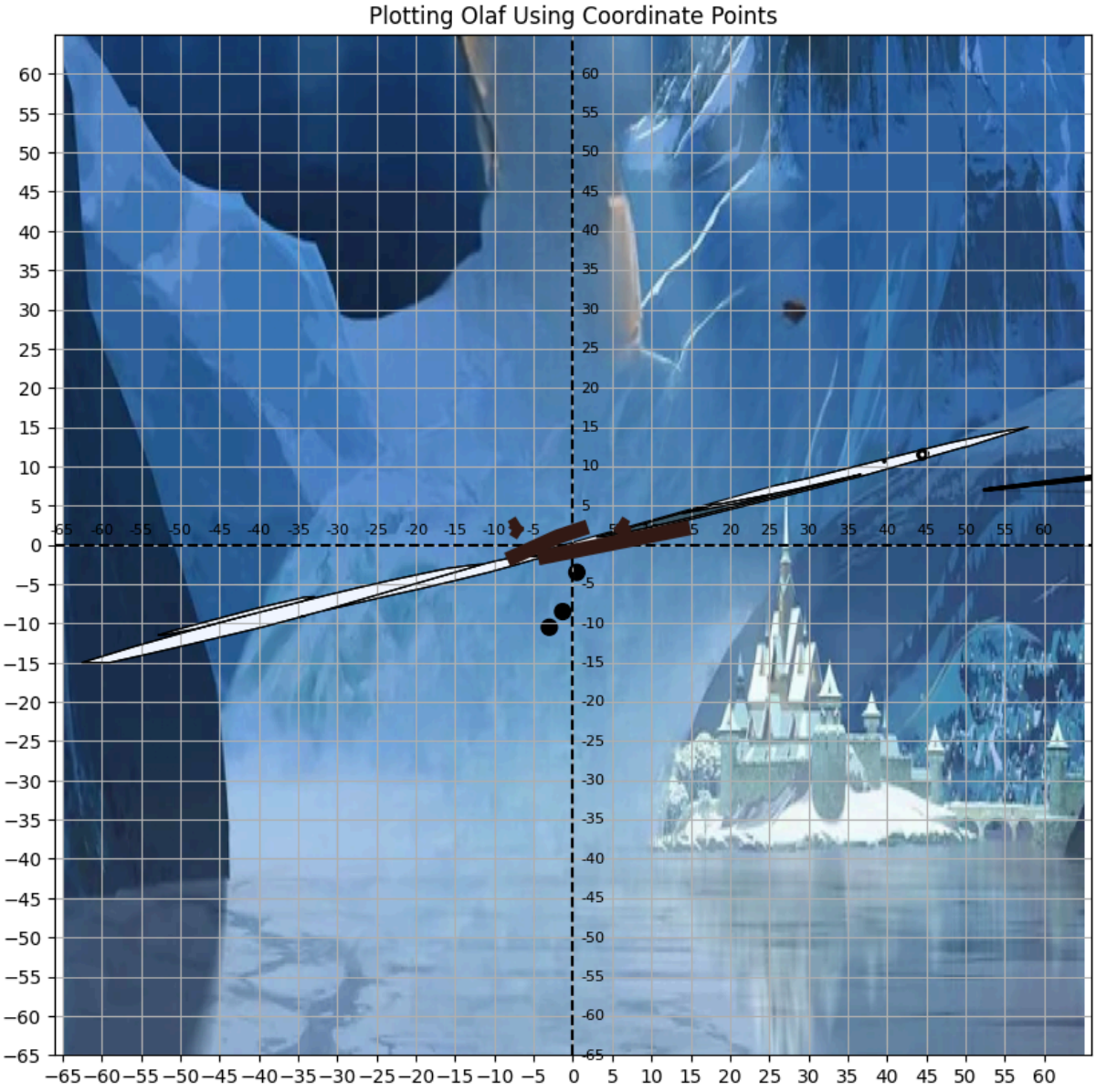
In the above problem, our K factor is 4. We simply have to multiply the x-coordinate by factor k, and add it to y coordinate. We simply define a formula which takes in the x and y coordinate lists, multiply each y coordinate point with 4 and add it to its respective x coordinate point. We apply this for all the lists

In [19]:
```
bg_img = mpimg.imread('olaf bg.png')
fig, ax = plt.subplots(figsize=(10, 15))

def apply_shear(x_coord:list, y_coord:list):
    sheared_x_coord = []
    for i in range(len(x_coord)):
        sheared_x_coord.append(x_coord[i] + (4*y_coord[i]))

    return sheared_x_coord

def single_shear(x_coord,y_coord):
    return x_coord + (4*y_coord)

ax.imshow(bg_img, extent=[-65, 65, -65, 65], zorder=0)

# head
x_coord = [-0.8, -0.6,   0,    1,    2,    2,    2,  1.7, 0,     0.3,   -2,
y_coord = [-0.8, -1.2, -1.3, -1.4,   -1,    0,    7,    9, 12.5,   12.4,   1
x_sheared_coord = apply_shear(x_coord, y_coord)

plt.fill(x_sheared_coord, y_coord, color='#f0f8ff', edgecolor='black', linew
plt.plot(x_sheared_coord, y_coord, color='black', linewidth=0.8)

plt.axis('equal')
plt.axhline(0, color='black', linestyle='--')
plt.axvline(0, color='black', linestyle='--')
plt.xticks(range(-65, 65, 5))
plt.yticks(range(-65, 65, 5))
plt.xlim(-66, 66)
plt.ylim(-65, 65)
plt.grid(True)


for i in range(-65, 65):
    if i != 0 and i % 5 == 0:

        ax.text(i, 1.0, str(i), ha='center', va='bottom', color='black', for

        ax.text(1.0, i, str(i), ha='left', va='center', color='black', fonts
plt.gca().set_aspect('equal', adjustable='box')


# eyes
eyex, eyey = (-1.5, 11.5)
```

```
x_shear_eye = single_shear(eyex, eyey)
eye1 = patches.Circle((x_shear_eye, eyey), radius=0.55, edgecolor='black', f
eye2x, eye2y = (-1.5, 11.5)
x_shear_eye = single_shear(eye2x, eye2y)
eye2 = patches.Circle((x_shear_eye, eye2y), radius=0.5, edgecolor='black', f
ax.add_patch(eye1)
ax.add_patch(eye2)

pupil1x, pupil1y = (-1.45, 11.65)
sheared_x_pupil = single_shear(pupil1x, pupil1y)
pupil1 = patches.Circle((sheared_x_pupil, pupil1y), radius=0.18, edgecolor='

pupil2x, pupil2y = (-2.95, 10.65)
sheared_x_pupil2 = single_shear(pupil2x, pupil2y)
pupil2 = patches.Circle((sheared_x_pupil2, pupil2y), radius=0.18, edgecolor=
ax.add_patch(pupil1)
ax.add_patch(pupil2)

#eye highlights
highlight1x, highlight1y =(-1.4, 11.75)
sheared_hightlight1x = single_shear(highlight1x, highlight1y)
highlight1 = patches.Circle((sheared_hightlight1x, highlight1y), radius=0.05

highlight2x, highlight2y =(-2.9, 10.75)
sheared_hightlight2x = single_shear(highlight2x, highlight2y)
highlight2 = patches.Circle((sheared_hightlight2x, highlight1y), radius=0.05
ax.add_patch(highlight1)
ax.add_patch(highlight2)
#nose

x_nose = [-1.5, -3.0, -3.5, -3.8, -3.5, -3.0, -2.0, -1.2]
y_nose = [10.5, 10.0, 9.0, 7.5, 7.0, 7.5, 8.0, 9.0]
sheared_x_nose = apply_shear(x_nose, y_nose)

t = np.linspace(0, 1, len(x_nose))
t_smooth = np.linspace(0, 1, 200)
cs_x = interp1d(t, sheared_x_nose, kind='cubic')
cs_y = interp1d(t, y_nose, kind='cubic')
x_smooth = cs_x(t_smooth)
y_smooth = cs_y(t_smooth)

sheared_x_smooth = apply_shear(x_smooth, y_smooth)

plt.fill(sheared_x_smooth, y_smooth, color='#FD6A02', edgecolor='black', lir
plt.plot(sheared_x_smooth, y_smooth, color='black', linewidth=2, zorder=4)

shadow_x = np.linspace(min(x_smooth), max(x_smooth), 50)
shadow_y = np.linspace(min(y_smooth), min(y_smooth) - 0.3, 50)
sheared_shadom_x = apply_shear(shadow_x, shadow_y)

plt.fill_between(sheared_shadom_x, shadow_y , min(y_smooth), color='black',


#smile

x_smile = [-2.7, -1,0,0.7,1,1,0.7,0,-1,-2, -3.1]
```

```
y_smile = [ 4.8,  1,1,2  ,3,8,9  ,7,6 ,5.2,5.2]

sheared_x_smile = apply_shear(x_smile, y_smile)

plt.fill(sheared_x_smile, y_smile, color='#3e5a69', edgecolor='black', linew
plt.plot(sheared_x_smile, y_smile, color='black', linewidth=0.8)

#teeth
x_teeth = [-2.4 ,-2, -0.2,-0.1,-0.4]
y_teeth = [5    ,4, 5.3 , 6,6.5]
sheared_x_teeth = apply_shear(x_teeth, y_teeth)


plt.fill(sheared_x_teeth, y_teeth, color='#f0f8ff', edgecolor='black', linew
plt.plot(sheared_x_teeth, y_teeth, color='black', linewidth=0.8)


#chest
x_chest = [-0.8,-1.2,-1.5,-1.6,-1.6,0 , 1.8  ,2.5 ,3,3.2    , 3.3,3    ,2.5,
y_chest = [-0.8,-1.2,-2  ,-2  ,-2.5,-5.5,-8.4,-6.5,-5.5,-4.5,-3.5,-2.7,-1.7,
sheared_x_chest = apply_shear(x_chest, y_chest)
plt.fill(sheared_x_chest, y_chest, color='#f0f8ff', edgecolor='black', linew
plt.plot(sheared_x_chest, y_chest, color='black', linewidth=0.8)


#body
x_body = [-1.6,-2  ,-3  ,-4,-5  ,-5.2,-5.5,-5.7,-5.7,-5.5,-5.0,-4,-2.5,1,1.5
y_body = [-2.5,-2.5,-2.7,-3,-4.3,-4.6,-5.5,-6.5,-7.5,-9.5,-10.9,-13,-15,-15,
sheared_x_body = apply_shear(x_body, y_body)

plt.fill(sheared_x_body, y_body, color='#f0f8ff', edgecolor='black', linewic
plt.plot(sheared_x_body, y_body, color='black', linewidth=0.8)

#feet

x_foot = [-5.7, -6.5, -7.5, -7.8, -7.9, -7.7,-7.5,-7.1,-7   ,-6.8 ,-6.2 ,-5.
y_foot = [-7.2, -6.6, -6.7, -7.5, -8  , -9  ,-10 ,-11 ,-11.2,-11.5,-11.5,-10
sheared_x_foot = apply_shear(x_foot, y_foot)
plt.fill(sheared_x_foot, y_foot, color='#f0f8ff', edgecolor='black', linewic
plt.plot(sheared_x_foot, y_foot, color='black', linewidth=0.8)

#buttons

button1= patches.Circle((0.5, -3.5), radius=1, edgecolor='black', facecolor=
button2 = patches.Circle((-1.3, -8.5), radius=1, edgecolor='black', facecolo
button3 = patches.Circle((-3, -10.5), radius=1, edgecolor='black', facecolor
ax.add_patch(button1)
ax.add_patch(button2)
ax.add_patch(button3)

# Left arm
x_left_arm = [2.5, 5, 6]
y_left_arm = [-1.5, 1, 2]
sheared_left_x_arm = apply_shear(x_left_arm, y_left_arm)

ax.plot(sheared_left_x_arm, y_left_arm , color='#341d17', linewidth=8, zorde
```

```
ax.plot([6, 6.5], [2, 2.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([6, 6.8], [2, 1.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([6, 6.6], [2, 2.3], color='#341d17', linewidth=5, zorder=5)

#right arm
x_right_arm = [-1.5, -6, -7]
y_right_arm = [-1.5, 1, 2]
sheared_x_right_arm = apply_shear(x_right_arm, y_right_arm)
ax.plot(sheared_x_right_arm, y_right_arm, color='#341d17', linewidth=8, zorc

ax.plot([-7, -7.5], [2, 2.8], color='#341d17', linewidth=5, zorder=5)
ax.plot([-7, -7.3], [2, 1.5], color='#341d17', linewidth=5, zorder=5)
ax.plot([-7, -7.4], [2, 2.3], color='#341d17', linewidth=5, zorder=5)

plt.title('Plotting Olaf Using Coordinate Points')

plt.grid(True)
plt.savefig('final.png')
plt.show()
```



Conclusion

Hence, we were successful in manually plotting Olaf using coordinate points. We applied expansion along y axis and shear along x axis. On expansion, its y coordinates expanded and the image was stretched along the y coordinate. The maximum coordinate was 60. So, we increased the grid limit from 15x15 to 60x60. On applying shear, its x coordinate was stretched along the x axis, the max coordinate along y coordinate was -65. so we increased the grid limit from 15x15 to 65x65