Задание: разработать анимированное изображение.

Математическая модель:

Код программы:

import tkinter as tk  
import numpy as np  
  
from converter import d2s, fig2s  
from converter import TRANSFORMATIONS as TR  
  
  
# Unit matrix  
UNIT\_MATRIX = np.array([  
 [1, 0],  
 [0, 1]  
])  
# Matrix reflect at y-axis  
TRY = np.array(TR['ry'])  
  
  
class Figure:  
 def \_\_init\_\_(self, symmetric, type\_, points, color, animated=False, t=None):  
 self.symmetric = symmetric  
 self.type = type\_  
 self.points = points  
 self.color = color  
 self.animated = animated  
 self.t = t  
  
 def switch(self):  
 if self.animated:  
 # switch biases  
 self.t = [-axis for axis in self.t]  
 self.points = [  
 [axis + bias for axis, bias in zip(row, self.t)]  
 for row in self.points  
 ]  
 # t = [[(lambda a: -a if a != 1 else a)(axis) for axis in row] for row in t]  
  
  
# Image  
# ORDER IS IMPORTANT!  
FIGURES = [  
 # Background  
 Figure(True, 'oval', [[1, -15.5], [13, -10]], '#c48157'), # Legs  
 Figure(False, 'line', [[17, 0], [17, 10]], 'black', True, [0, 0.8]), # Rope  
 Figure(False, 'oval', [[14, 9], [20, 14]], 'red', True, [0, 0.5]), # Balloon  
 Figure(True, 'oval', [[2.5, -2.5], [18, 4]], '#c48157', True, [0, 0.8]), # Hands  
 Figure(True, 'oval', [[2, 18.5], [10, 11]], '#84d3db'), # Ears  
 # Front  
 Figure(True, 'oval', [[-10, -15], [10, 15]], '#84d3db'), # Body  
 Figure(True, 'oval', [[0.5, 8], [2, 3]], 'white'), # Teeth  
 Figure(True, 'oval', [[-3, 8], [3, 5]], '#c9996c'), # mouth  
 Figure(True, 'oval', [[-2, 9], [2, 7]], 'black'), # Nose  
 Figure(True, 'oval', [[0.5, 10], [4, 13.5]], 'white'), # Eyes  
 Figure(True, 'oval', [[1, 11], [1.5, 11.5]], 'black', True, [-0.5, 0]), # Eyebrows  
]  
  
  
def main():  
 def draw():  
 canvas.delete('all')  
  
 for fig in FIGURES:  
 method = getattr(canvas, f'create\_{fig.type}')  
  
 m = np.array(fig.points)  
 t = UNIT\_MATRIX  
  
 # Move figure before reflect and operate it  
 # Else unit matrix will be used  
 if fig.animated:  
 fig.switch()  
  
 transformed = np.matmul(m, t)  
 method(\*fig2s(transformed), fill=fig.color)  
  
 # Draw again maybe reflected  
 if fig.symmetric:  
 t = TRY # Reflect at y  
  
 transformed = np.matmul(m, t)  
 method(\*fig2s(transformed), fill=fig.color)  
  
 canvas.after(500, draw)  
  
 draw()

Протокол работы программы:

