|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 2**

|  |  |
| --- | --- |
| **Тема:** Преобразование изображений  **Студент:** Саркисов А. С.  **Группа:** ИУ7-43  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

Москва.

2020 г.

**Цель работы**

Познакомиться с назначением, областями применения, сущностью операций преобразования; научиться разрабатывать программы, осуществляющие преобразования изображений на плоскости.

Изображение выглядит как доска, текст

Автоматически созданное описание

**Техническое задание**

Изобразить исходный рисунок и реализовать возможность выполненять над ним преобразования(перенос, масштабирование и поворот).

**Теоретическая часть**

**Перенос**

Перенос изображения заключается в перемещении отображенного объекта из одного места экрана в другое место. Перенос изображения позволяет построить рисунок в произвольном месте экрана и затем перенести его в другую, требуемую, часть экрана. При этом можно изменить компоновку рисунка или создать единый рисунок из набора готовых элементов.

Для переноса точки из позиции с координатами (X,Y) в позицию с координатами (X1,Y1) надо к координате X добавить DX, а к координате Y- DY единиц, причем DX = X1 - X, DY = Y1 - Y.

Таким образом, X1 = X + DX, Y1 = Y + DY.

Положительное значение DX означает перемещение точки вправо по горизонтали, отрицательное - влево; положительное значение DY - перемещение вниз по вертикали, отрицательное - вверх.

Также следует отметить, что необходимо задавать такие значения DX, DY, чтобы после преобразования точка оставалась в пределах экрана, иначе она не будет отображаться на нем. Не следует задавать слишком малые значения DX, DY (DX < 0.5; DY < 0.5), так как в этом случае точка повторно высвечивается на старом месте.

Перенос рисунка из одной области экрана в другую эквивалентен переносу всех точек рисунка и последующему высвечиванию соединяющих их линий. Для исключения искажения рисунка все точки необходимо переместить на одинаковое расстояние.

**Масштабирование изображения**

При создании изображения на экране дисплея может возникнуть необходимость изменения его размеров с целью повышения его наглядности, для вставки созданного изображения в уже имеющийся рисунок. Размер рисунка можно изменить, если умножить все расстояния между точками на некоторую постоянную величину (коэффициент масштабирования). Если коэффициент масштабирования больше единицы, то рисунок увеличивается, если меньше единицы - уменьшается.

Наряду с коэффициентом масштабирования для выполнения масштабирования надо указать новое положение рисунка (после выполнения масштабирования). Новое положение рисунка определяется центром масштабирования - некоторой центральной точкой, относительно которой выполняется масштабирование.

Матрица преобразования при масштабировании имеет вид:

| KX 0 0 |

М = | 0 KY 0 |

| 0 0 1 |

Поэтому для точек применимо:

X1 = X \* KX, Y1 = Y \* KY,

где KX - коэффициент масштабирования по оси абсцисс;

KY - коэффициент масштабирования по оси ординат.

Таким образом, применив данную формулу ко всем точкам рисунка, получается изображение, промасштабированное по центру координат.

Для того, чтобы промасштабировать точку по заданному центру, следует воспользоваться следующими формулами:

X1=X \* KX + (1 - KX) \* XM

Y1=Y \* KY + (1 - KY) \* YM

**Поворот изображения**

Для выполнения поворота надо указать величину угла, на который необходимо осуществить поворот, и координаты точки, которая берется за центр вращения. Если исходную точку A с координатами (X,Y) по дуге окружности с центром в точке C с координатами (Xс,Yс) поворачивают на угол t, то координаты (X1,Y1) повернутой точки могут быть записаны в следующем виде:

X1 = Xс + (X - Xс) \* cos(t) + (Y - Yс) \* sin(t)

Y1 = Yс + (Y - Yс) \* cos(t) - (X - Xс) \* sin(t)

Если начало координат расположено в левой верхней точке экрана, то угол поворота измеряется в направлении против часовой стрелки. Если же начало координат лежит в левой нижней точке экрана, то угол поворота должен измеряться в направлении по часовой стрелке.

**Коммутативность преобразований**

В ряде частных случаев наблюдается коммутативность операций преобразования. В таблице приведены варианты таких преобразований:

|  |  |
| --- | --- |
| M1 | M2 |
| Перенос  Масштабирование  Поворот  Масштабирование (однородное) | Перенос  Масштабирование  Поворот  Поворот |

**Практическая часть**

**dataOptions** - словарь, хранящий значения, для работы с преобразованиями.

dataOptions = {

"center": (0.0, 1.0),

"scale": [1.0, 1.0],

"sumScale": [1.0, 1.0],

"angle": 90,

"singlePiece": 350.0

}

**Перенос:**

def transpose(placeGraph, listPoints, dataOptions, dx, dy):

placeGraph.delete("all")

singlePiece = dataOptions["singlePiece"]

center = (

dataOptions["center"][0] \* dataOptions["sumScale"][0],

dataOptions["center"][1] \* dataOptions["sumScale"][1]

)

x, y = getСoordPoint(center, singlePiece, center)

paintPoint(placeGraph, x, y, "blue")

printPoint(placeGraph, center, singlePiece, center, text=dataOptions["center"])

*for* i in range(len(listPoints)):

listPoints[i][0] += dx

listPoints[i][1] += dy

createGraphByPoints(placeGraph, listPoints, singlePiece, center)

**Масштабирование:**

def scale(placeGraph, listPoints, dataOptions):

placeGraph.delete("all")

scaleX, scaleY = dataOptions["scale"][0], dataOptions["scale"][1]

singlePiece = dataOptions["singlePiece"]

*#dataOptions["realCenter"] = (dataOptions["realCenter"][0] \* scaleX, dataOptions["realCenter"][1] \* scaleY)*

center = (

dataOptions["center"][0] \* dataOptions["sumScale"][0],

dataOptions["center"][1] \* dataOptions["sumScale"][1]

)

print("center:", center)

x, y = getСoordPoint(center, singlePiece, center)

paintPoint(placeGraph, x, y, "blue")

printPoint(placeGraph, center, singlePiece, center, text=dataOptions["center"])

*for* i in range(len(listPoints)):

*#print(listPoints[i], end=' ')*

listPoints[i][0] \*= scaleX

listPoints[i][1] \*= scaleY

*#print(listPoints[i])*

createGraphByPoints(placeGraph, listPoints, singlePiece, center)

**Поворот:**

def rotation(placeGraph, listPoints, dataOptions):

placeGraph.delete("all")

singlePiece = dataOptions["singlePiece"]

rot = dataOptions["angle"]

center = (

dataOptions["center"][0] \* dataOptions["sumScale"][0],

dataOptions["center"][1] \* dataOptions["sumScale"][1]

)

x, y = getСoordPoint(center, singlePiece, center)

paintPoint(placeGraph, x, y, "blue")

printPoint(placeGraph, center, singlePiece, center, text=dataOptions["center"])

print("\n\n1 list:", listPoints)

*for* i in range(len(listPoints)):

x = center[0] + (listPoints[i][0] - center[0]) \* math.cos(math.radians(rot)) -\

(listPoints[i][1] - center[1]) \* math.sin(math.radians(rot))

y = center[1] + (listPoints[i][1] - center[1]) \* math.cos(math.radians(rot)) +\

(listPoints[i][0] - center[0]) \* math.sin(math.radians(rot))

listPoints[i] = [x, y]

print("\n\n2 list:", listPoints)

print("\n\n", listPoints, "\n\n", center)

createGraphByPoints(placeGraph, listPoints, singlePiece, center)

**Пользовательский интерфейс (UI):**

При запуске программы строится исходный рисунок. Если необходимо вернуться к первоначальному рисунку после ряда преобразований, необходимо воспользоваться кнопкой «BACK TO ORIGINAL».

Изображение выглядит как снимок экрана

Автоматически созданное описание

**Операция переноса**

Смещение на (0, 5):

Изображение выглядит как снимок экрана

Автоматически созданное описание

Смещение ещё на (-15.5, 10):

Изображение выглядит как снимок экрана

Автоматически созданное описание

Возвращение к первоначальной позиции с помощью смещения на (15.5, 10):

Изображение выглядит как снимок экрана

Автоматически созданное описание

Обработка ошибочного ввода пользователем:

Изображение выглядит как снимок экрана

Автоматически созданное описание

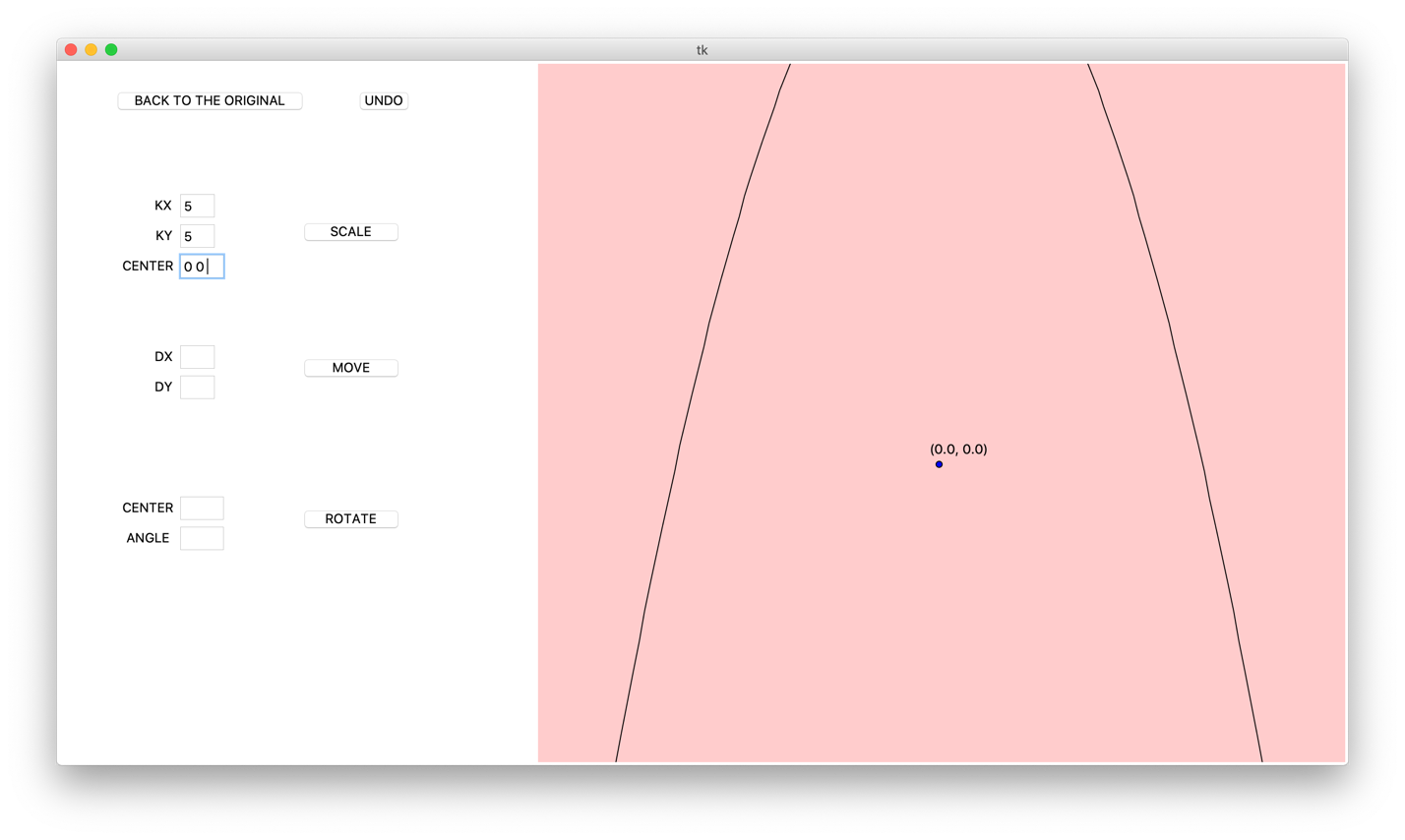
**Операция масштабирования относительно точки:** **Изображение выглядит как снимок экрана, монитор, экран

Автоматически созданное описание**Изображение выглядит как снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана

Автоматически созданное описание



Изображение выглядит как снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана

Автоматически созданное описание

Обработка ошибочного ввода пользователем:

Изображение выглядит как снимок экрана

Автоматически созданное описаниеИзображение выглядит как снимок экрана

Автоматически созданное описание

**Операция поворота относительно заданной точки:**

**Изображение выглядит как снимок экрана, монитор, экран

Автоматически созданное описание**

Изображение выглядит как снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана

Автоматически созданное описание

В данной части программы также предусмотрены ошибочные ситуации ввода:

Изображение выглядит как снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана

Автоматически созданное описание

**Выполним несколько не коммутативных преобразований:**

Выполним масштабирование, перенос, обратное масштабирование, обратный перенос.

Центр = (0.0, 0.0)

1. Уменьшим изображение в 2 раза

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Перенесем на (10, 0)

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Увеличим в 2 раза

Изображение выглядит как снимок экрана

Автоматически созданное описание

1. Перенесем на (-10, 0)

Изображение выглядит как снимок экрана

Автоматически созданное описание

**Вывод**: Текущая позиция отличается от первоначальной, т.к. перенос и масштабирование не коммутативные преобразования.

Также в программе реализован возврат к предыдущим состояниям рисунка (нажать на кнопку UNDO один или несколько раз). Рассмотрим на примере приведенных мной поворотов (вернемся в первоначальное состояние).

**Изображение выглядит как снимок экрана, монитор, экран

Автоматически созданное описание**

Изображение выглядит как снимок экрана

Автоматически созданное описание

Изображение выглядит как снимок экрана

Автоматически созданное описание

Если до текущего состояния рисунка других состояний нет.

Изображение выглядит как снимок экрана

Автоматически созданное описание

**Механизм хранения предыдущего состояния**

После каждого преобразования в специальный список записывается важные данные для обратного преобразования. Каждый элемент списка - это преобразование, которое было выполнено в порядке соответствующему номеру в списке. При шаге назад берется последний элемент списка и выполняется обратное преобразование. В каждом элементе хранится к какому типу относится данная операция (rotation, scale, transpose). После выполнения преобразования, последний элемент списка удаляется. Это возможно пока в списке будет хотя бы одно преобразование.

**Реализация шага назад:**

def setUndo(placeGraph, widgets, dataOptions):

*try*:

el = widgets["returnData"][-1]

*if* el[0] == "scale":

dataOptions["scale"] = el[1]["lastScale"]

dataOptions["center"] = el[1]["lastCenter"]

dataOptions["sumScale"] = el[1]["sumScale"]

scale(placeGraph, widgets["listPoints"], dataOptions)

*elif* el[0] == "rotation":

dataOptions["angle"] = el[1]["lastAngle"]

rotation(placeGraph, widgets["listPoints"], dataOptions)

placeGraph.delete("all")

dataOptions["center"] = el[1]["lastCenter"]

createGraphByPoints(placeGraph, widgets["listPoints"], dataOptions["singlePiece"], dataOptions["center"])

center = (

dataOptions["center"][0] \* dataOptions["sumScale"][0],

dataOptions["center"][1] \* dataOptions["sumScale"][1]

)

x, y = getСoordPoint(center, dataOptions["singlePiece"], center)

paintPoint(placeGraph, x, y, "blue")

printPoint(placeGraph, center, dataOptions["singlePiece"], center, text=dataOptions["center"])

*elif* el[0] == "traspose":

dx = -el[1]["dx"]

dy = -el[1]["dy"]

transpose(placeGraph, widgets["listPoints"], dataOptions, dx, dy)

widgets["returnData"].pop()

*except*:

messagebox.showinfo(title="Nothing undo.", message="All possible steps back have been taken.")

*return*