|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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:** Якуба Д. В.  **Группа:** ИУ7-43  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |  |

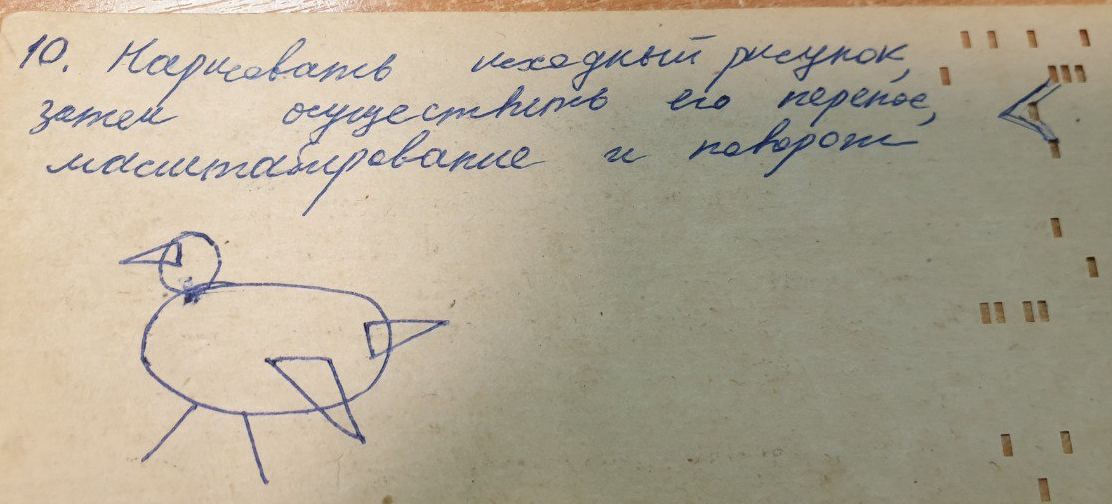
Москва.

2020 г.

**Цель работы**

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

**Техническое задание**



Изобразить исходный рисунок и реализовать возможность выполнения над ним преобразований: 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 высвечиваться не будет. 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.

**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.

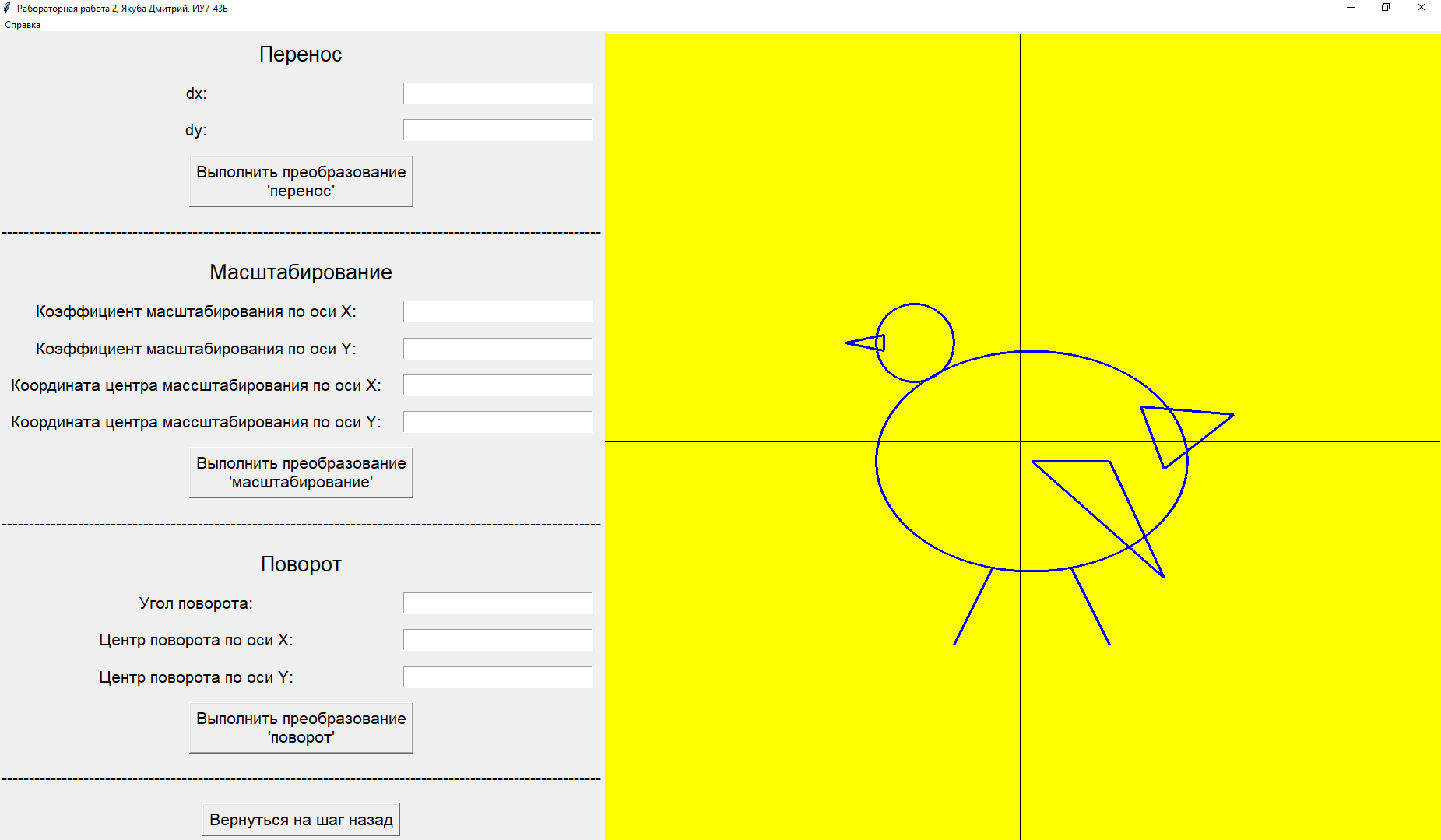
**Коммутативность преобразований**

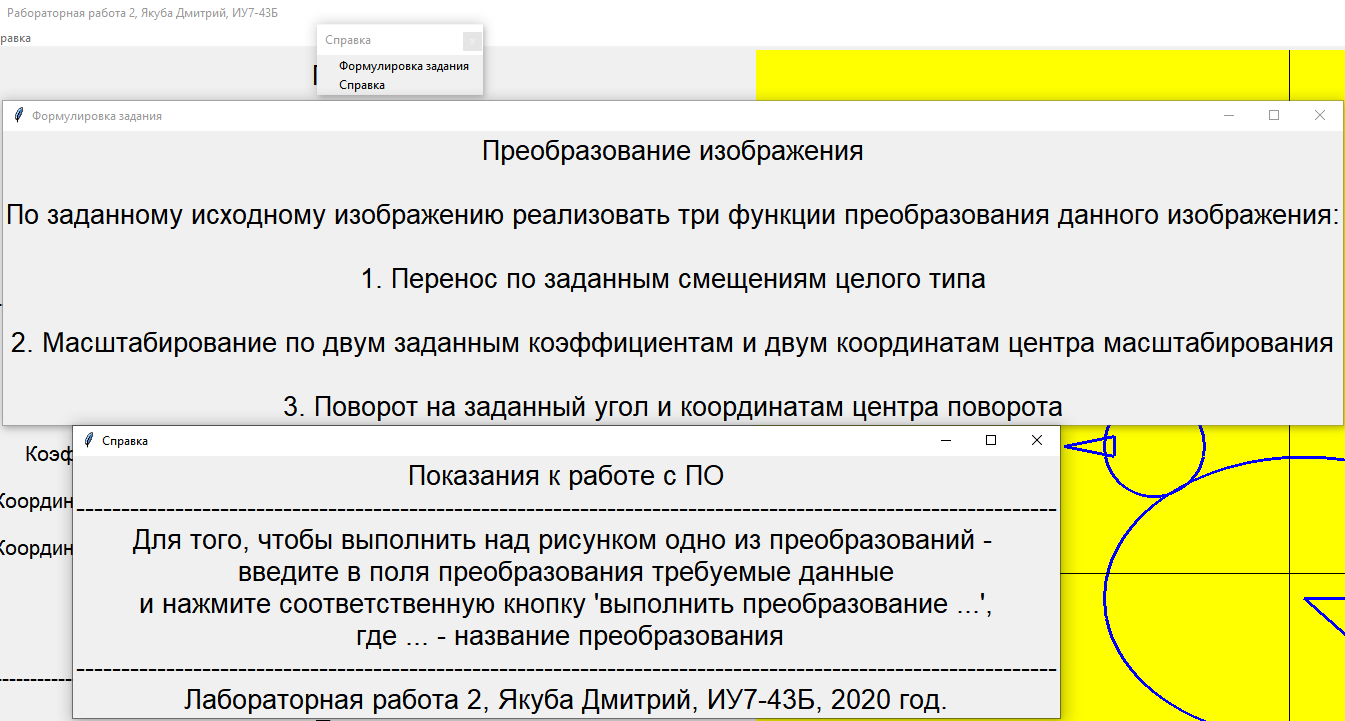
В ряде частных случаев наблюдается коммутативность операций преобразования. В таблице приведены варианты таких преобразований:

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

**Практическая часть**

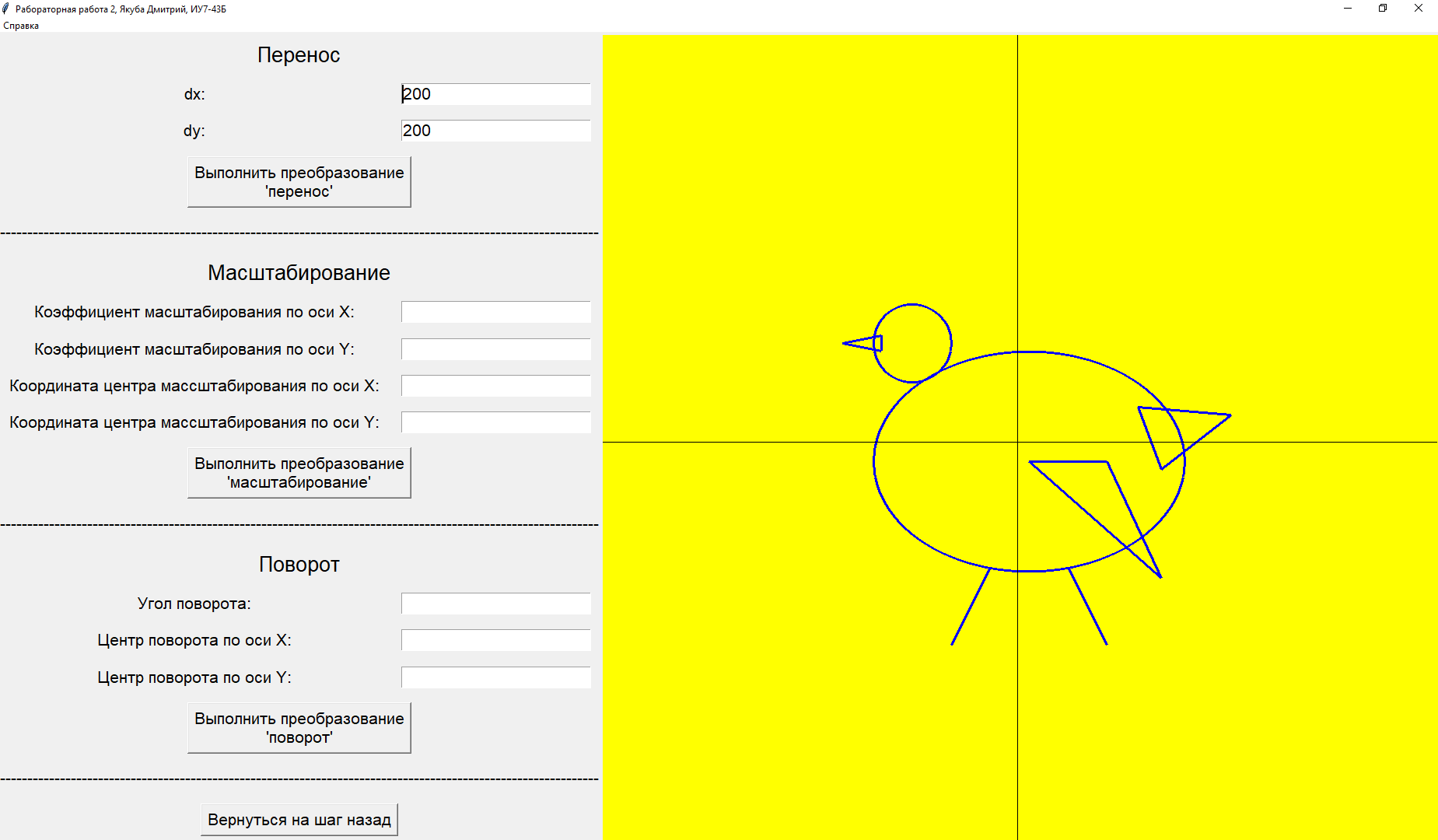
**Пользовательский интерфейс**

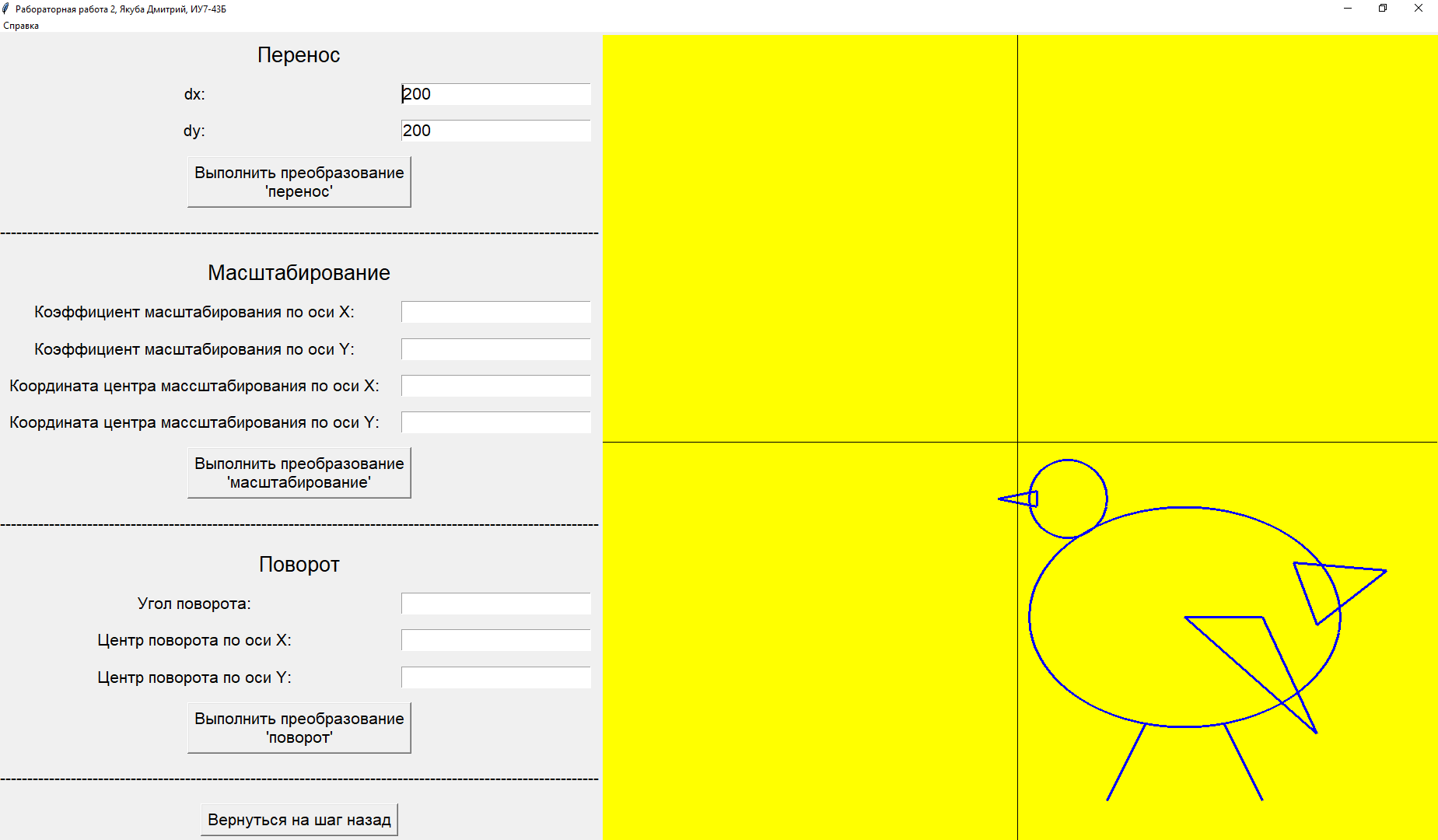


****

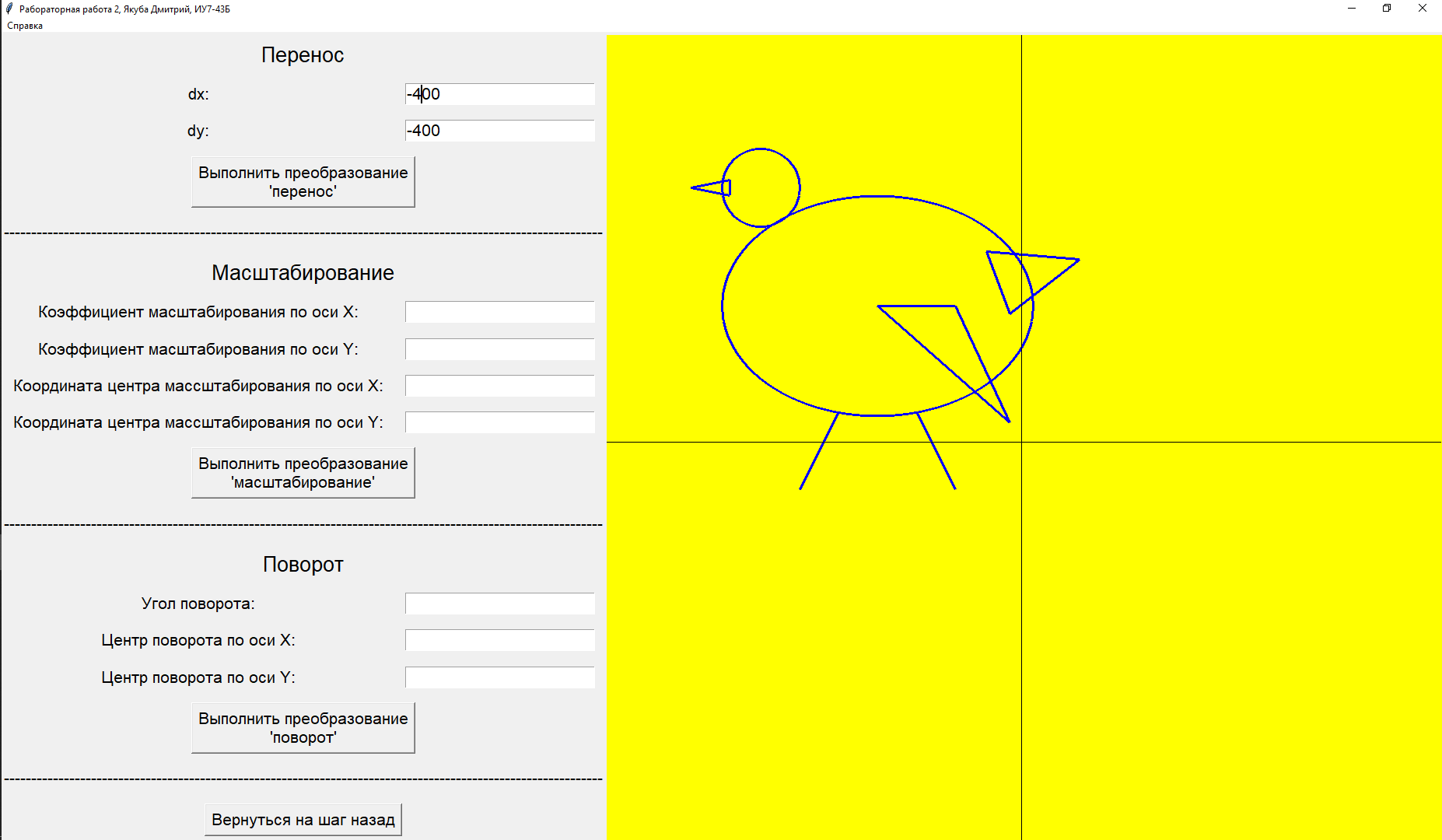
**Операция переноса**

Смещение на 200 200:

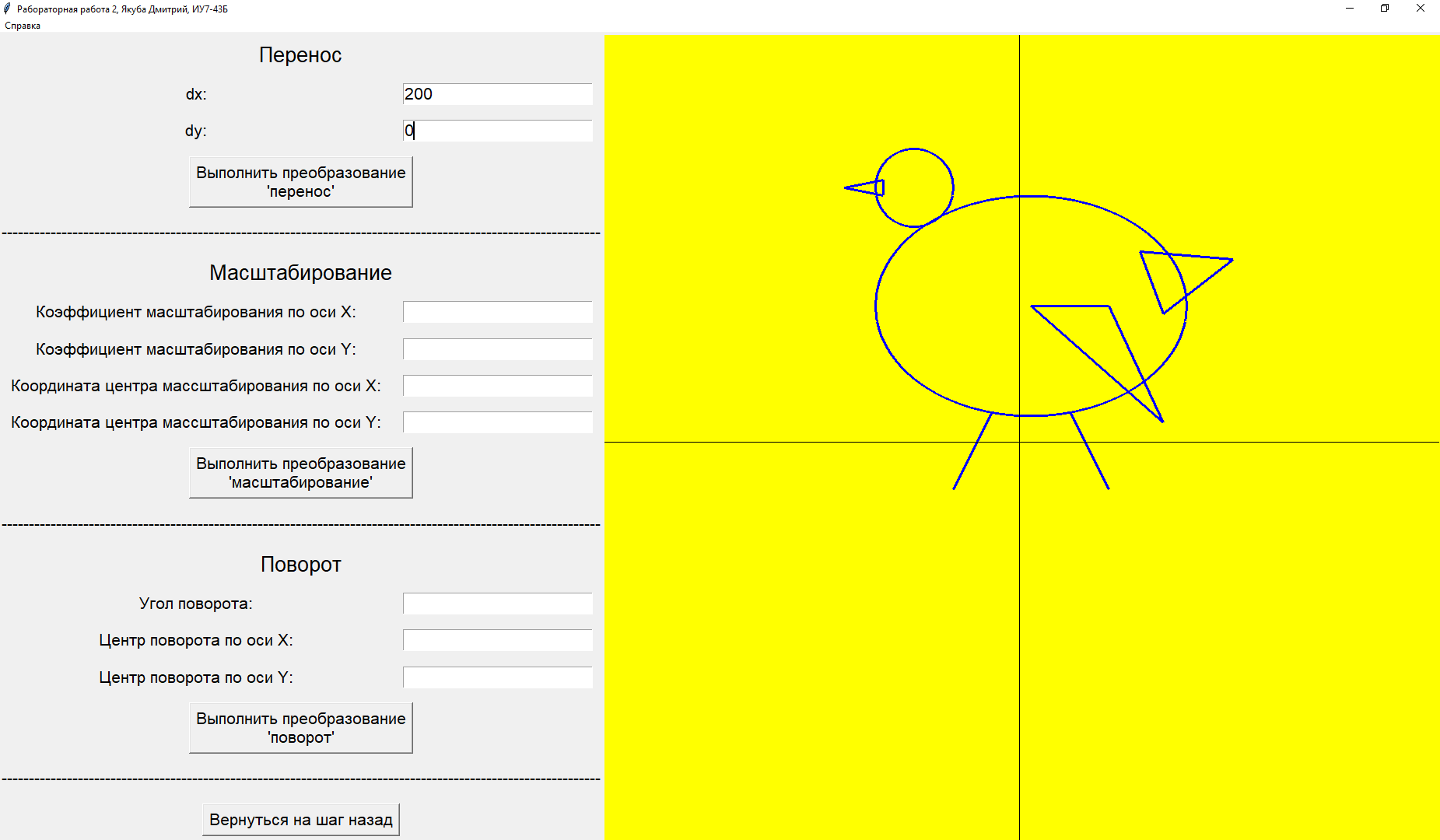




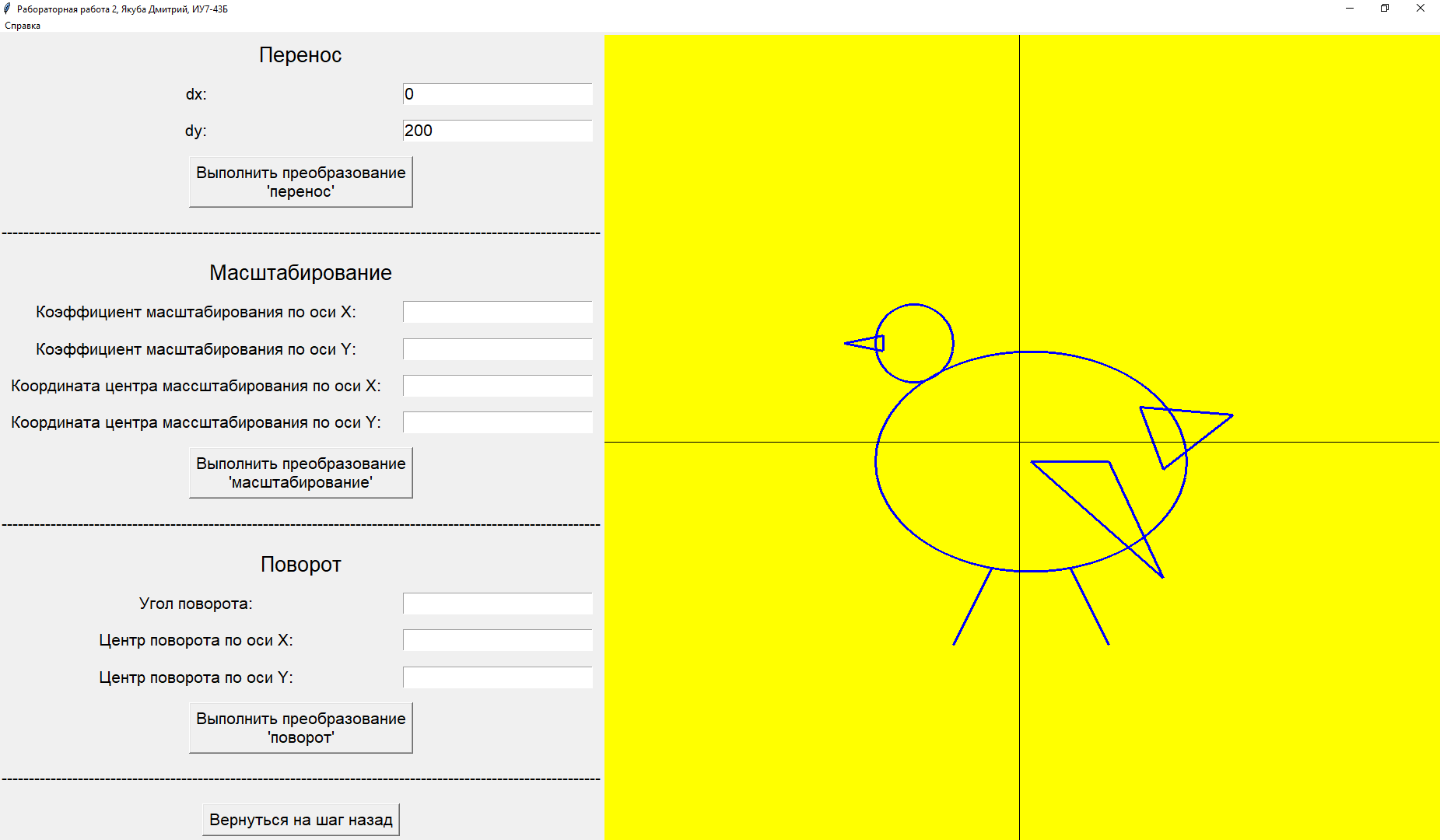
Смещение на -400 -400 из предыдущего состояния:



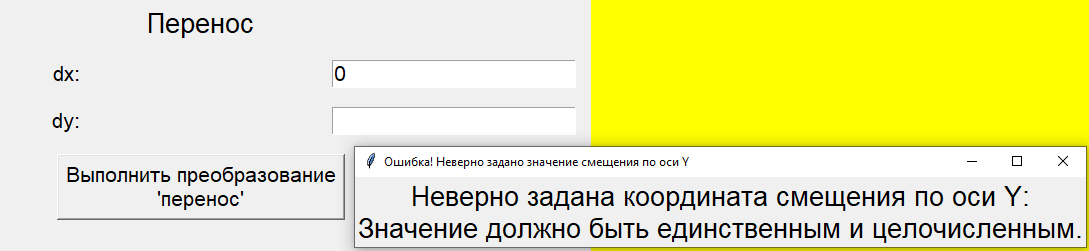
Смещение на 200 0 из текущего состояния:

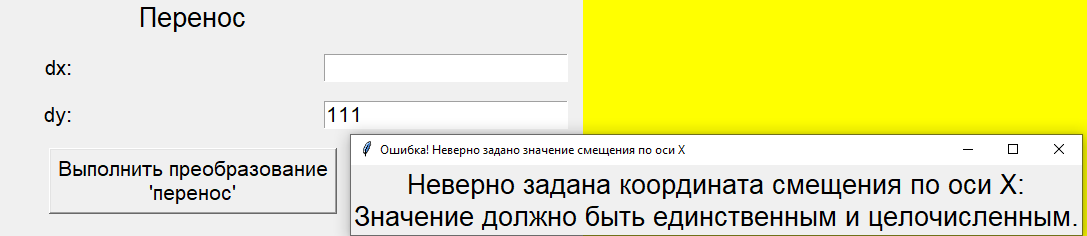


Возвращение к центру рисунка с помощью смещения на 0 200:



Отмечу, что интерфейсом предусмотрены ошибочные ситуации. Так, например, если оставить одно из полей смещений по осям пустым, ПО выдаст ошибку:

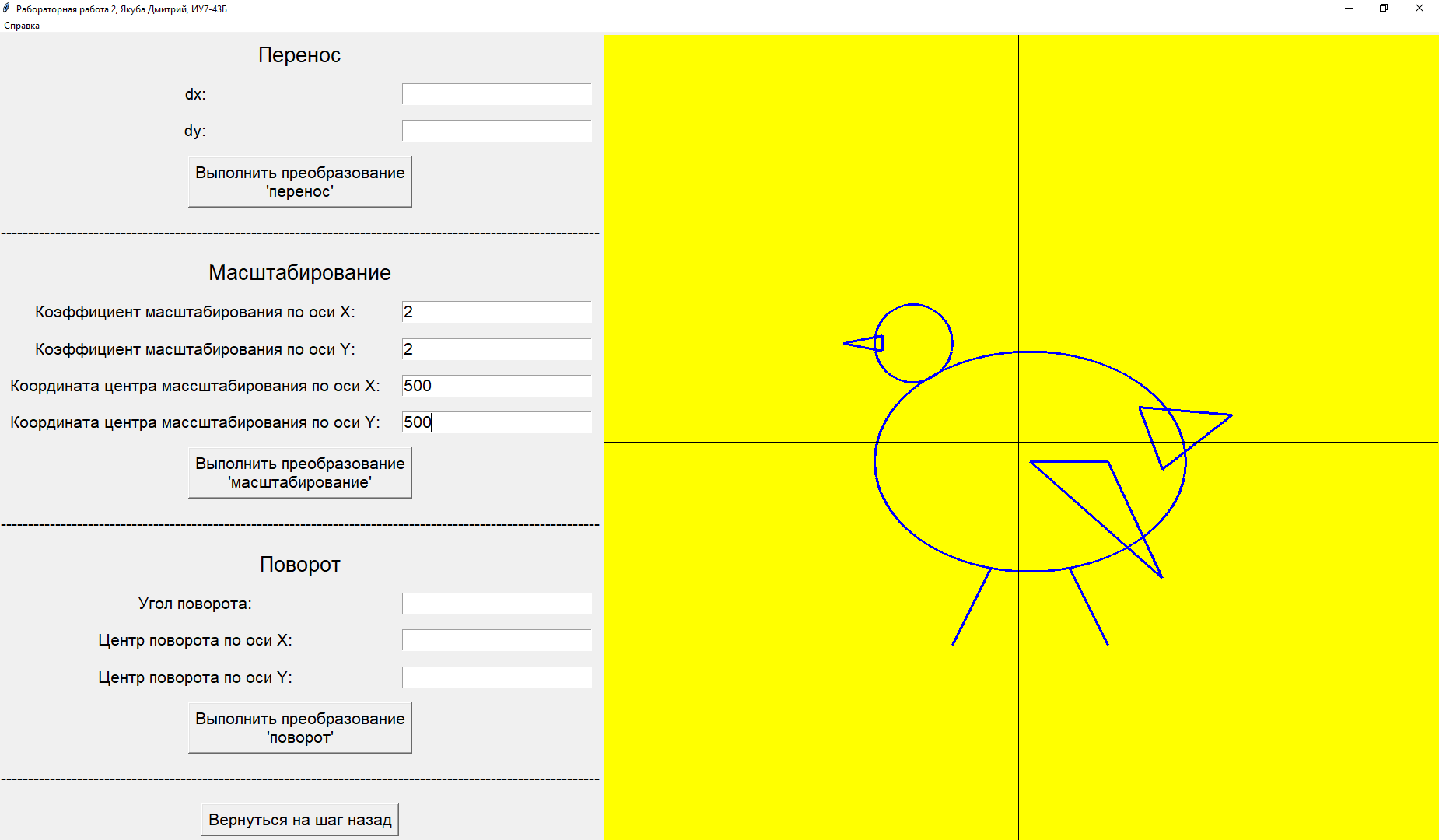


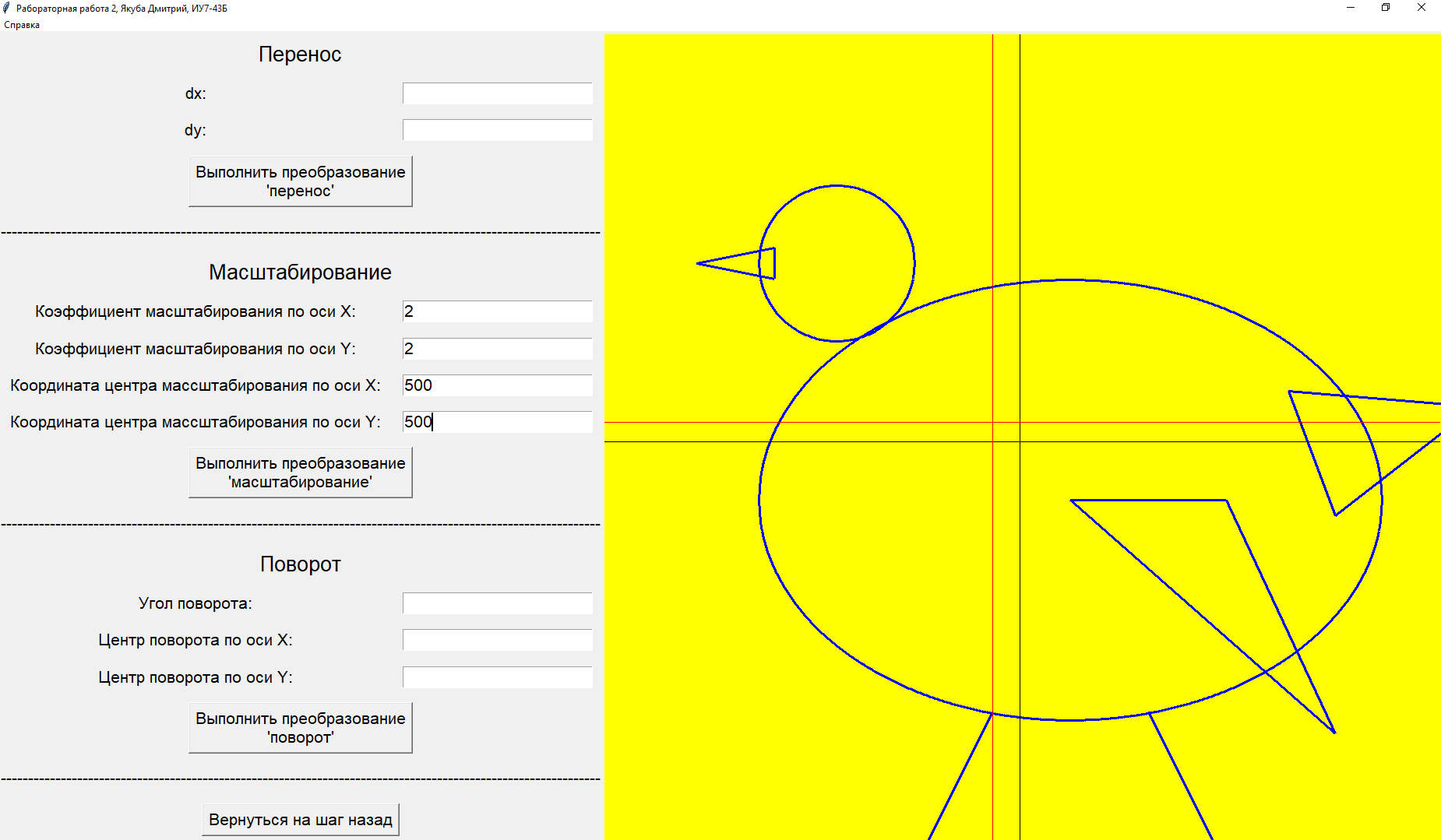


**Алгоритм:**

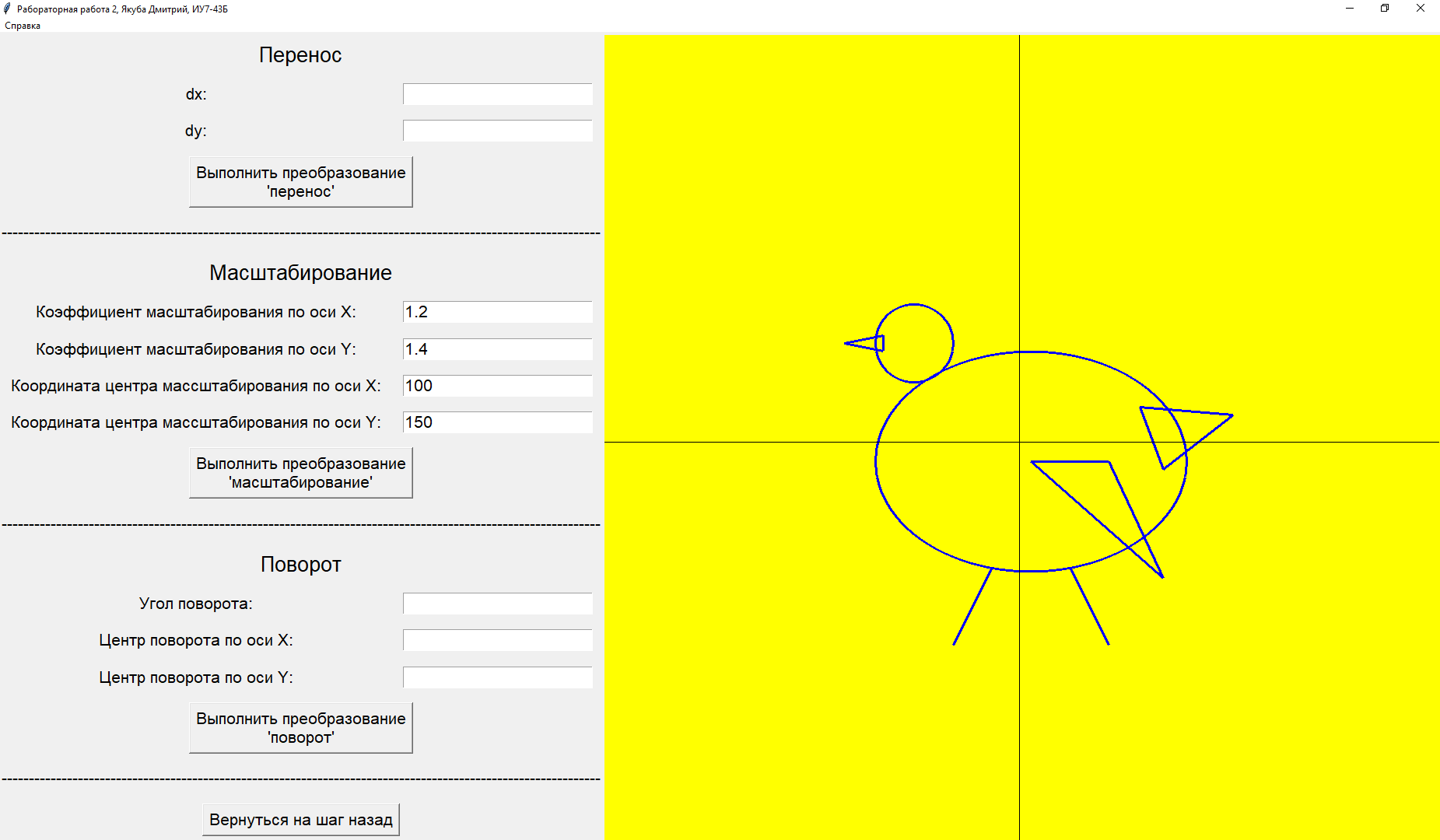
def transferArray(array, transferX, transferY):  
 for i in array:  
 i[0] += transferX  
 i[1] += transferY  
  
  
def transferImage(canvasWindow, entryX, entryY):  
 try:  
 xTransfer = entryX.get()  
 xTransfer = int(xTransfer)  
 except Exception:  
 makeErrorBadXTransfer()  
 return  
  
 try:  
 yTransfer = entryY.get()  
 yTransfer = int(yTransfer)  
 except Exception:  
 makeErrorBadYTransfer()  
 return  
  
 copyAllToPrev()  
  
 transferArray(dotsArrayTale, xTransfer, yTransfer)  
 transferArray(dotsArrayLeftLeg, xTransfer, yTransfer)  
 transferArray(dotsArrayMouth, xTransfer, yTransfer)  
 transferArray(dotsArrayHead, xTransfer, yTransfer)  
 transferArray(dotsArrayEll, xTransfer, yTransfer)  
 transferArray(dotsArrayRightLeg, xTransfer, yTransfer)  
 transferArray(dotsArrayWing, xTransfer, yTransfer)  
  
 printAll(canvasWindow)

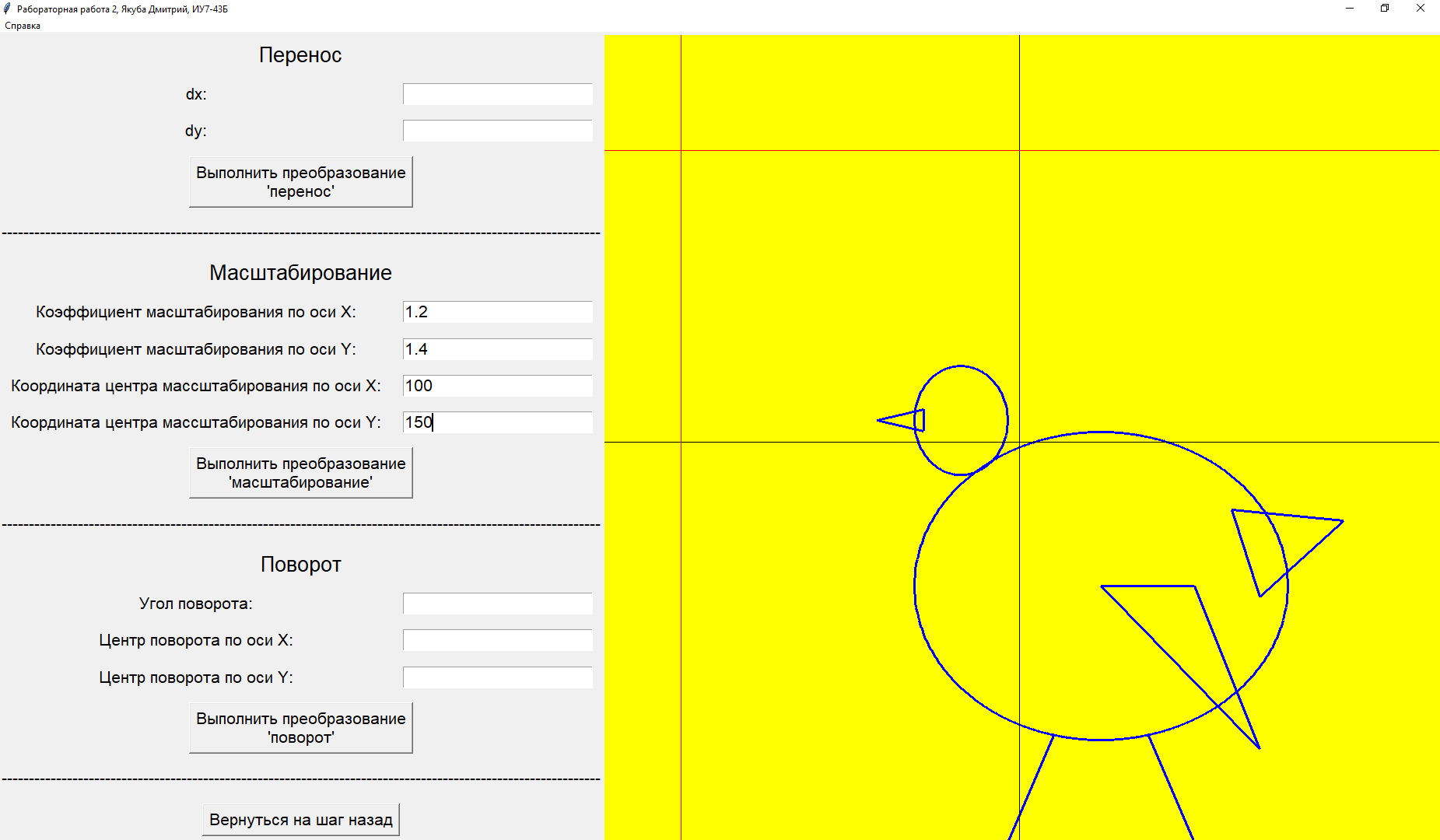
**Операция масштабирования относительно точки**

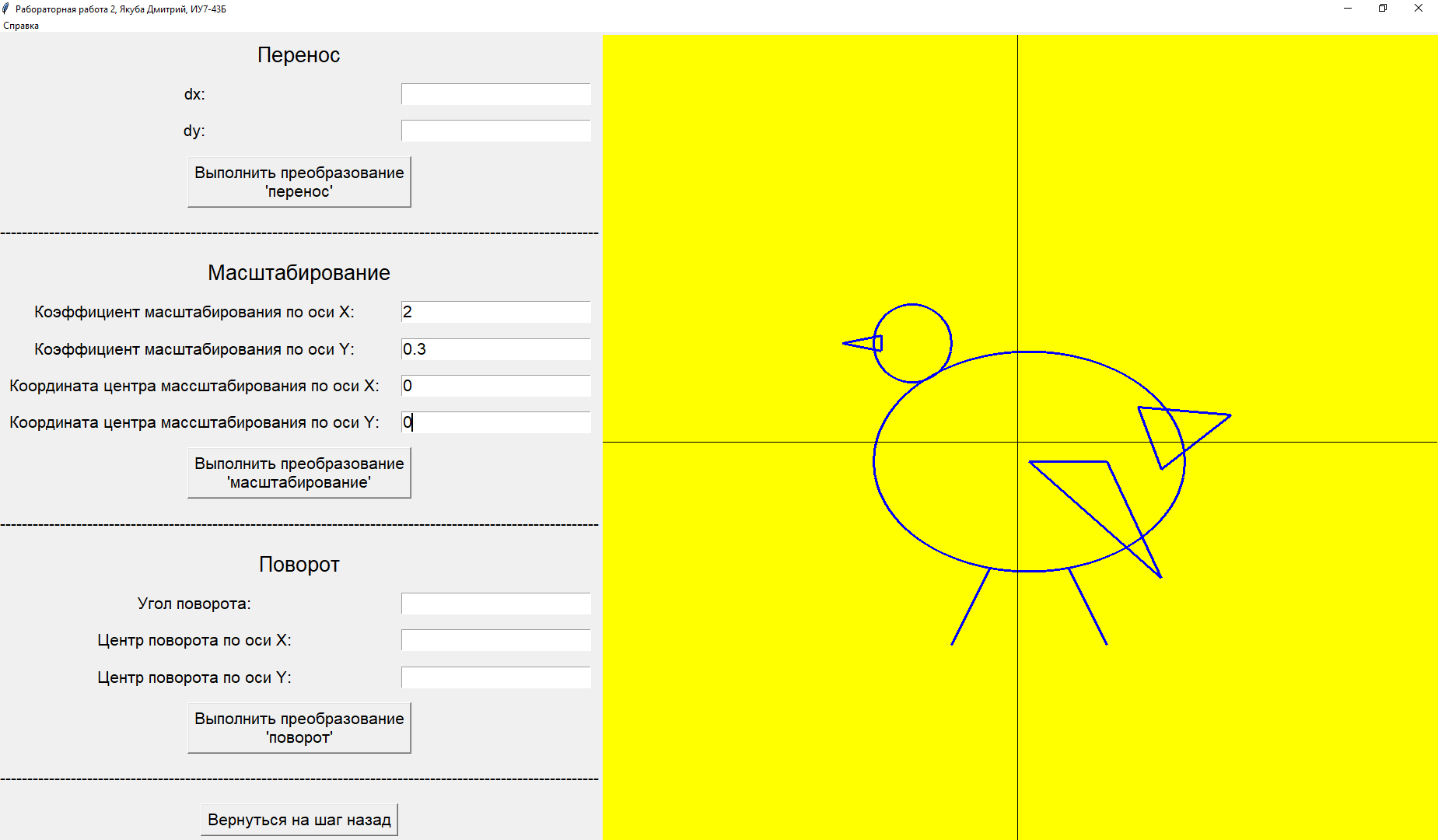


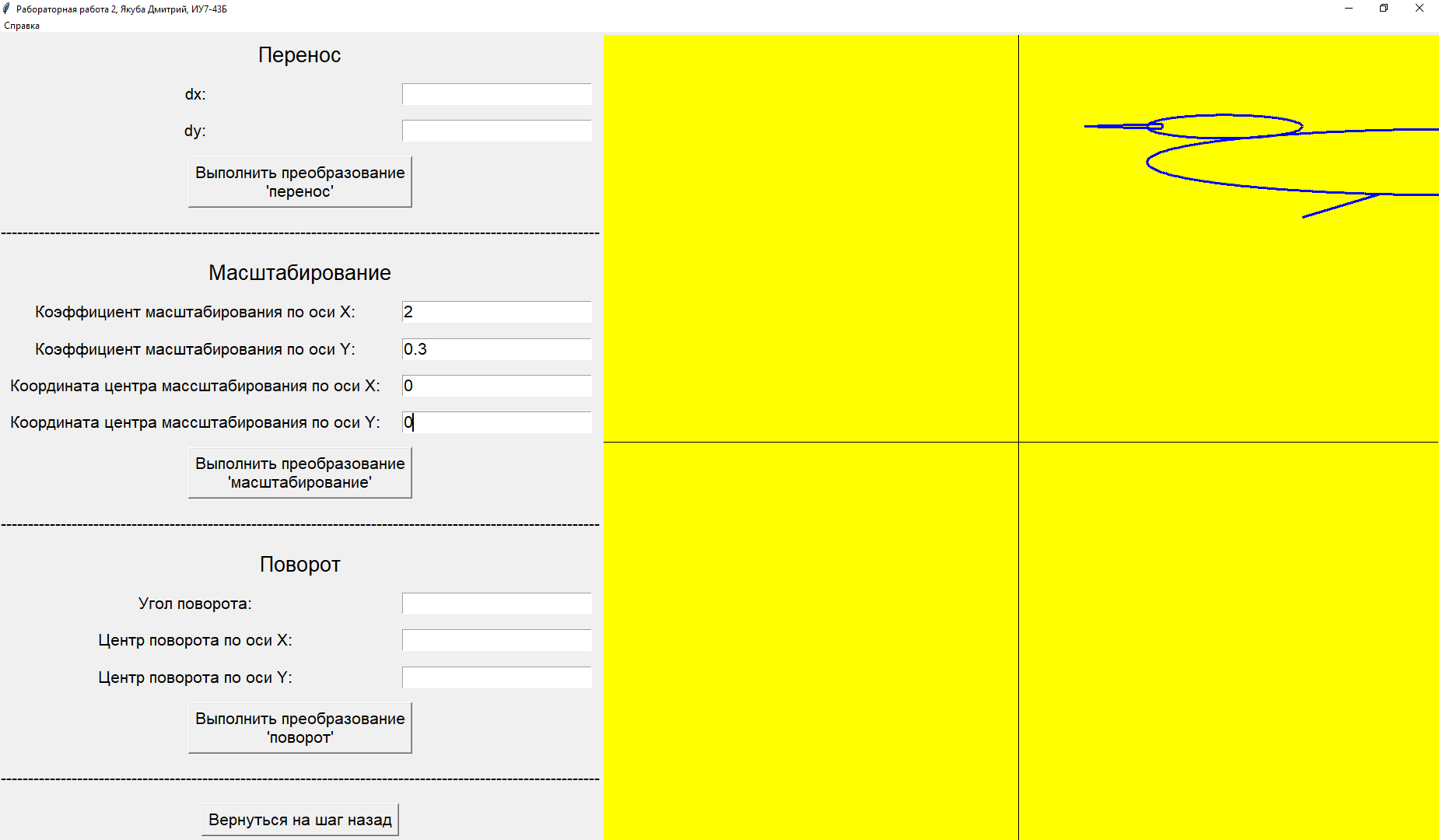


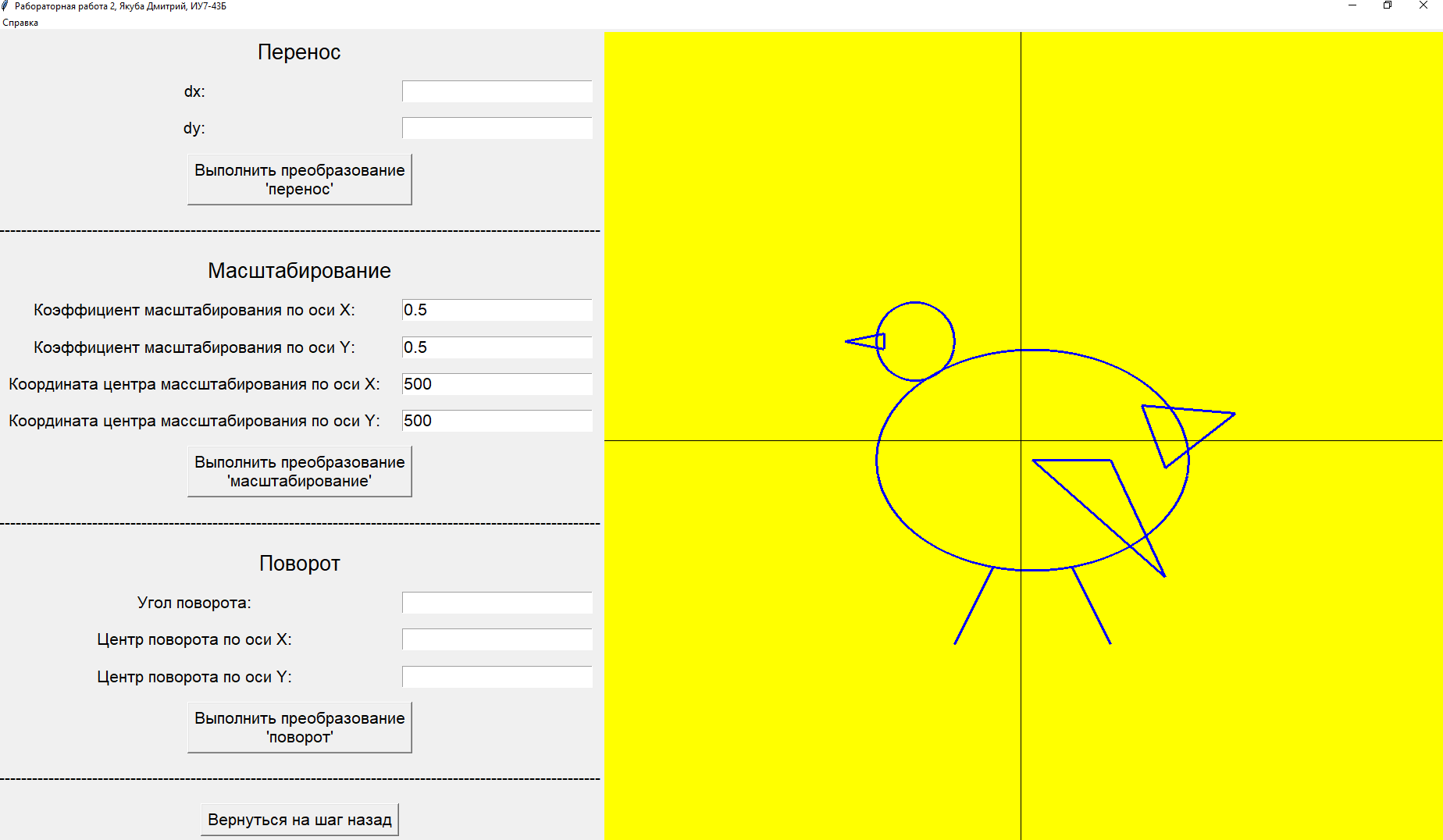
**Две красные перпендикулярные линии показывают центр масштабирования рисунка.**

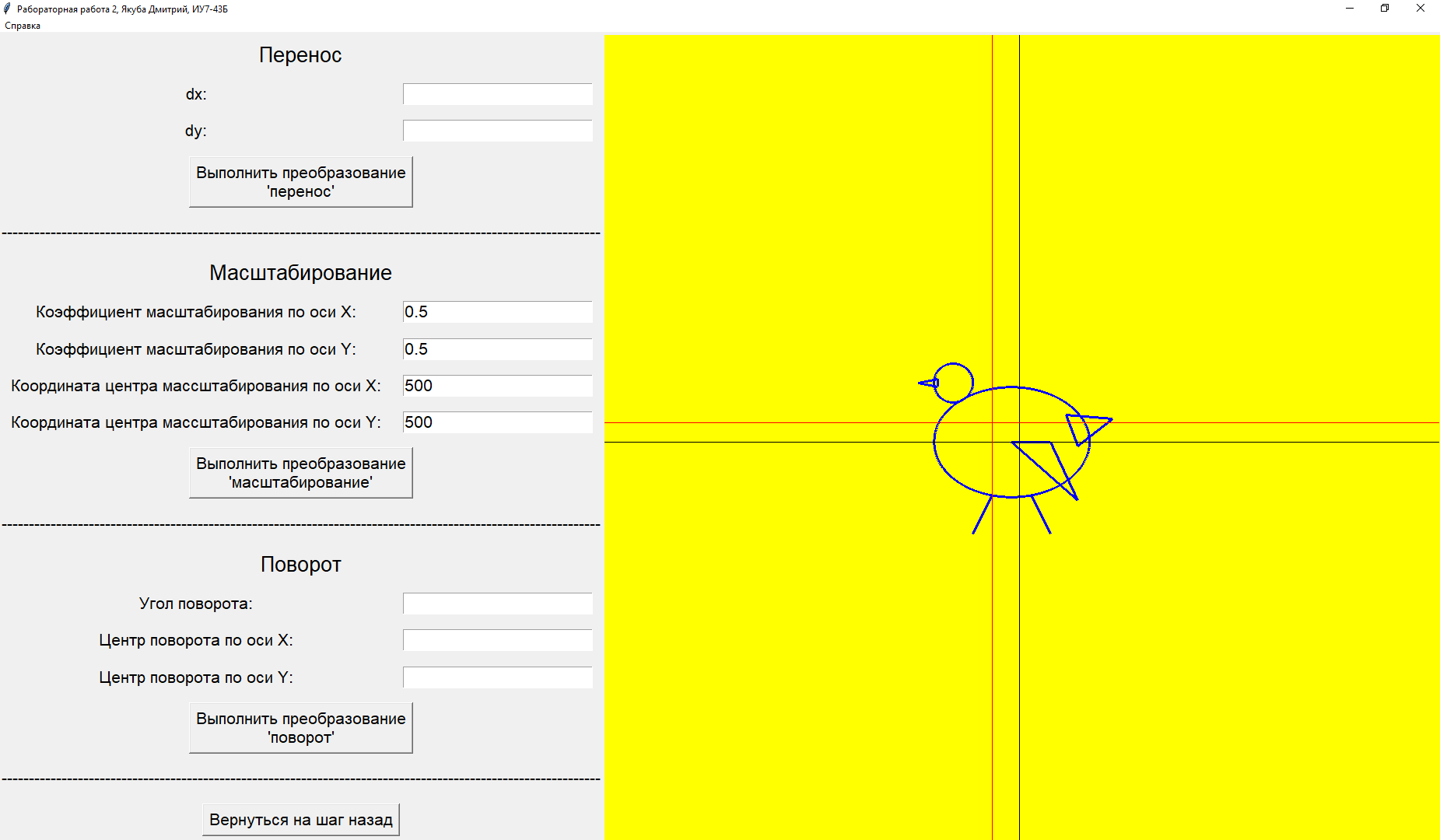


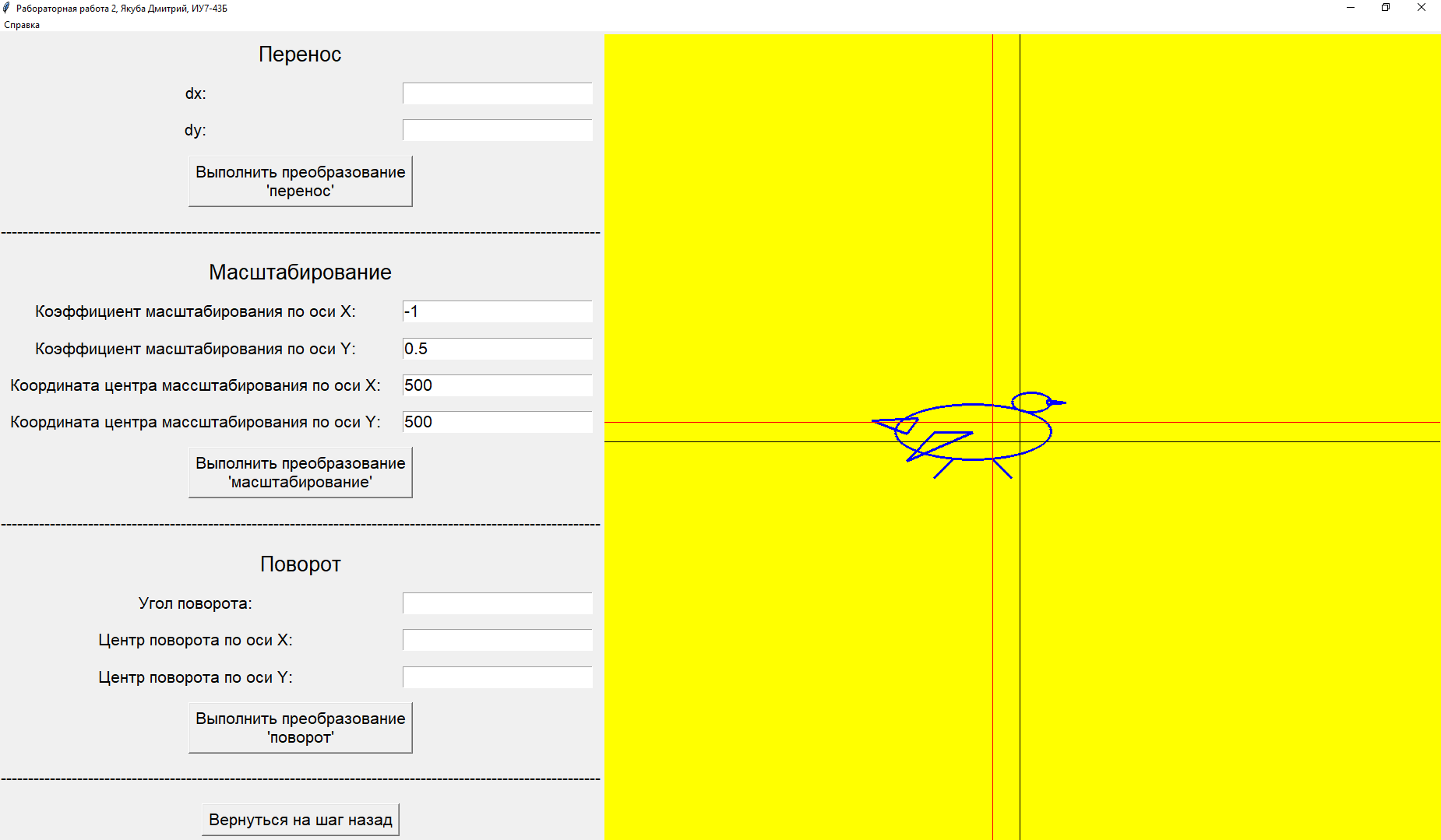


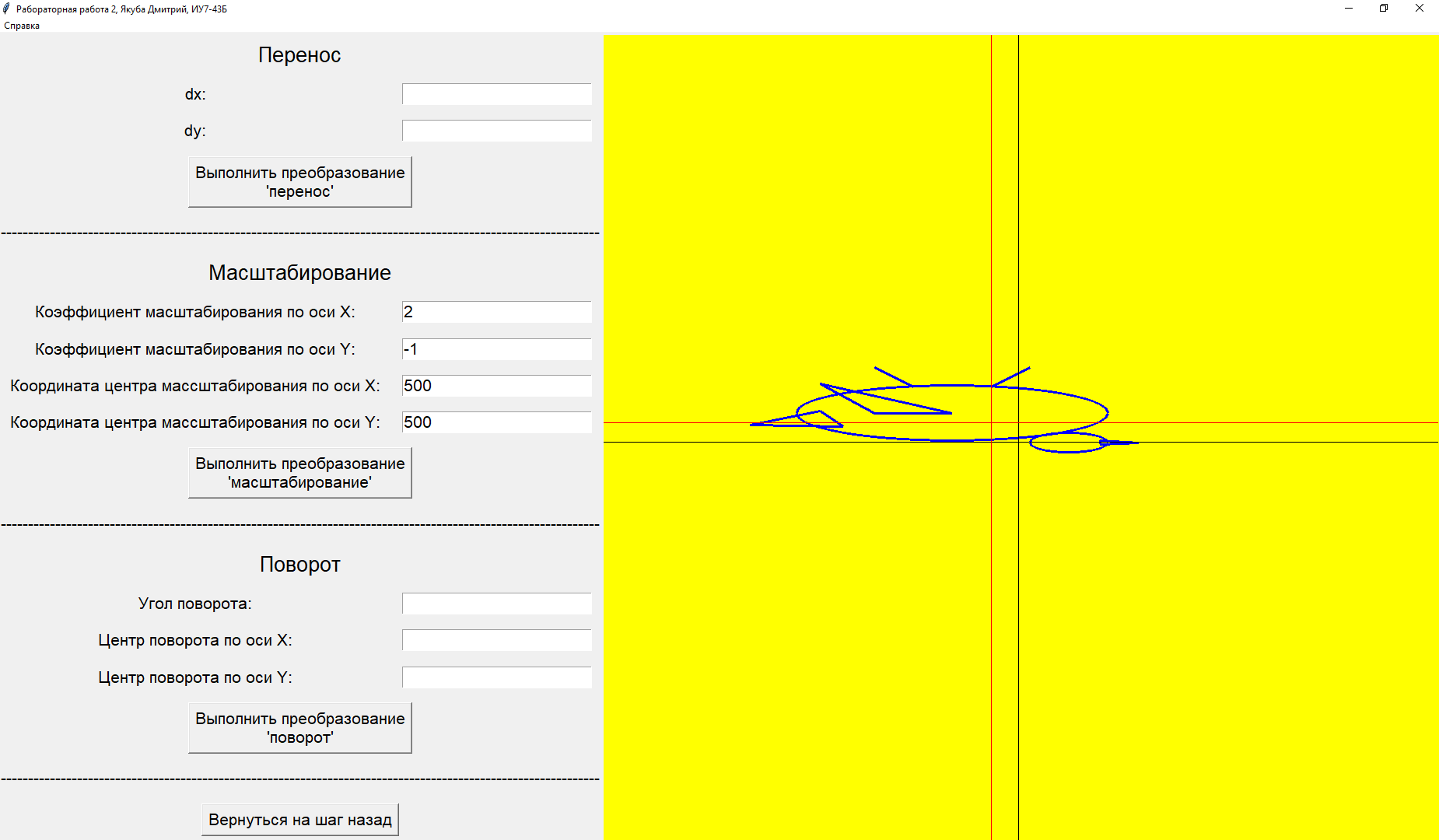




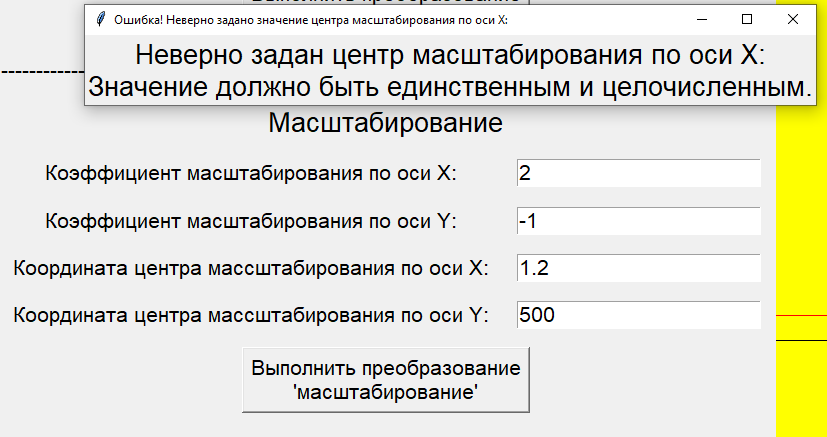


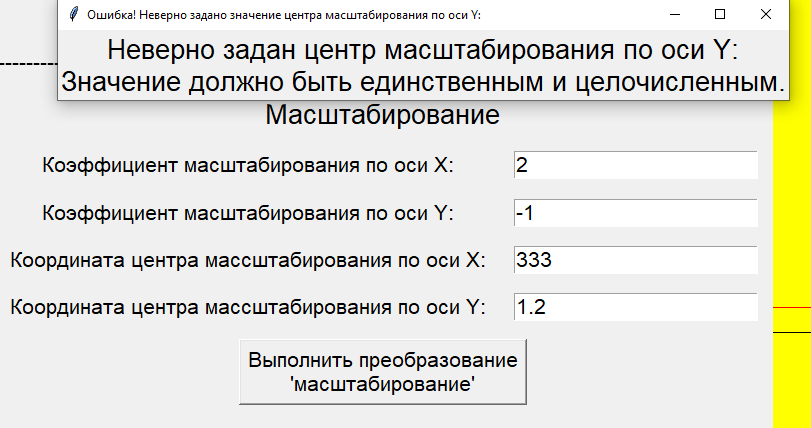


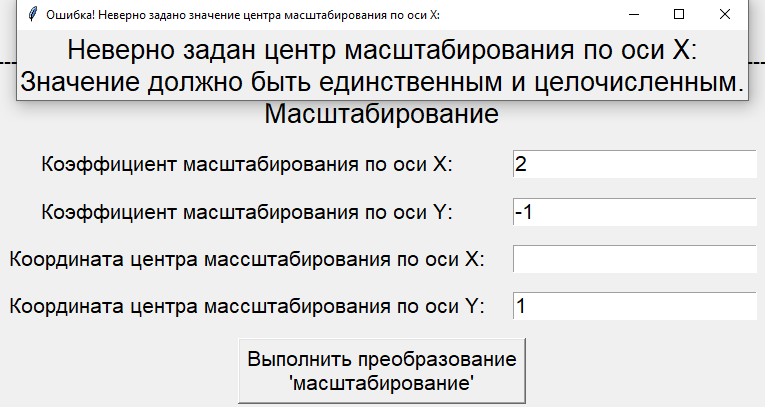


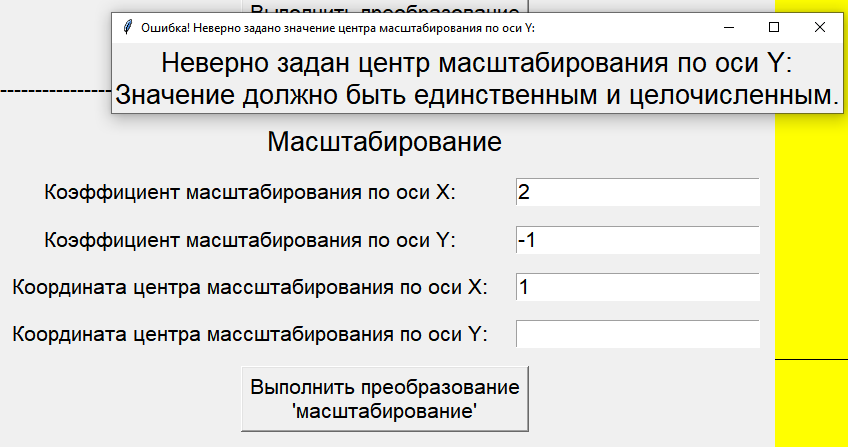


В данной части программы также предусмотрена проверка передаваемых пользователем данных:





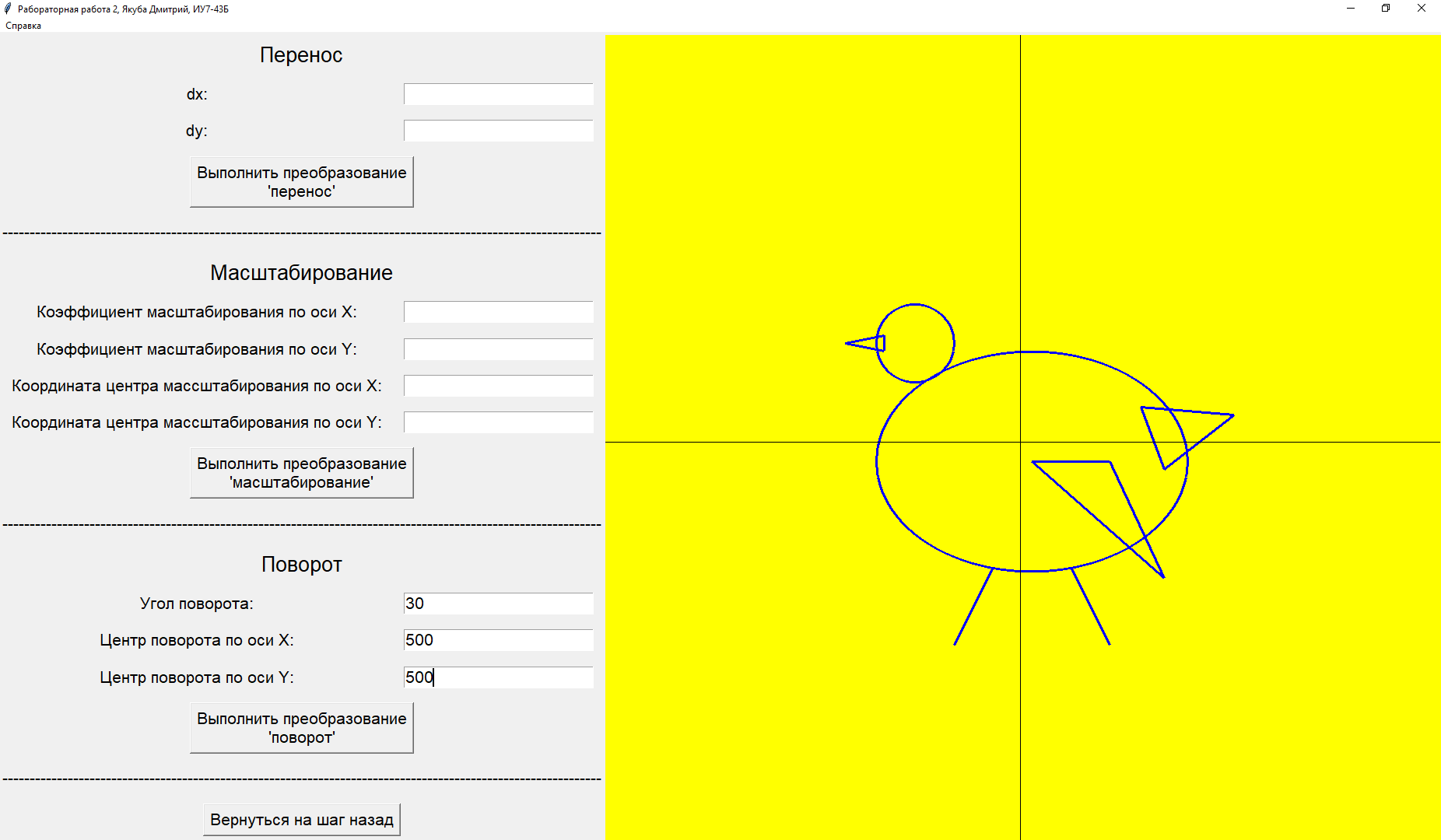


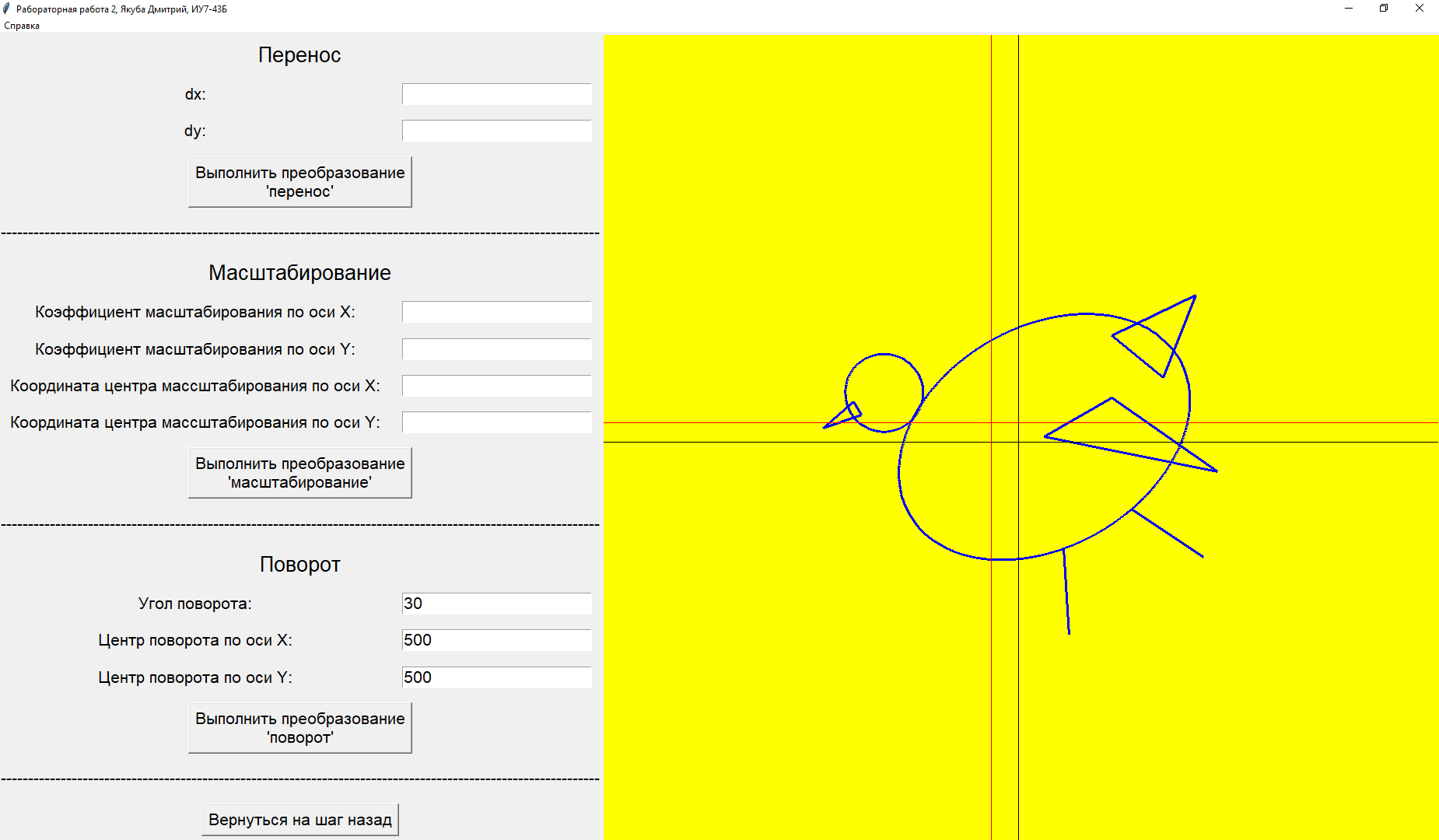


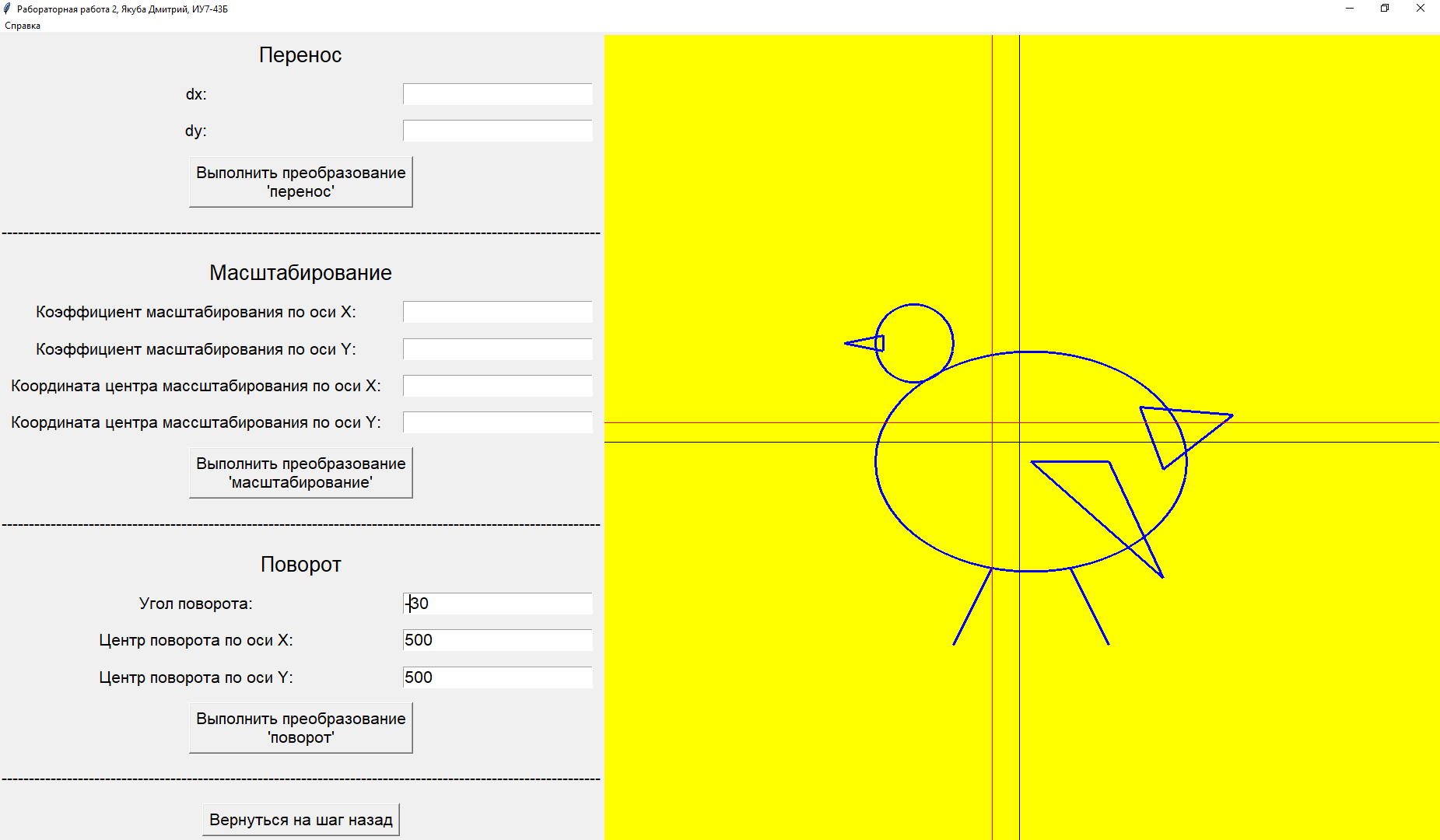
**Алгоритм:**

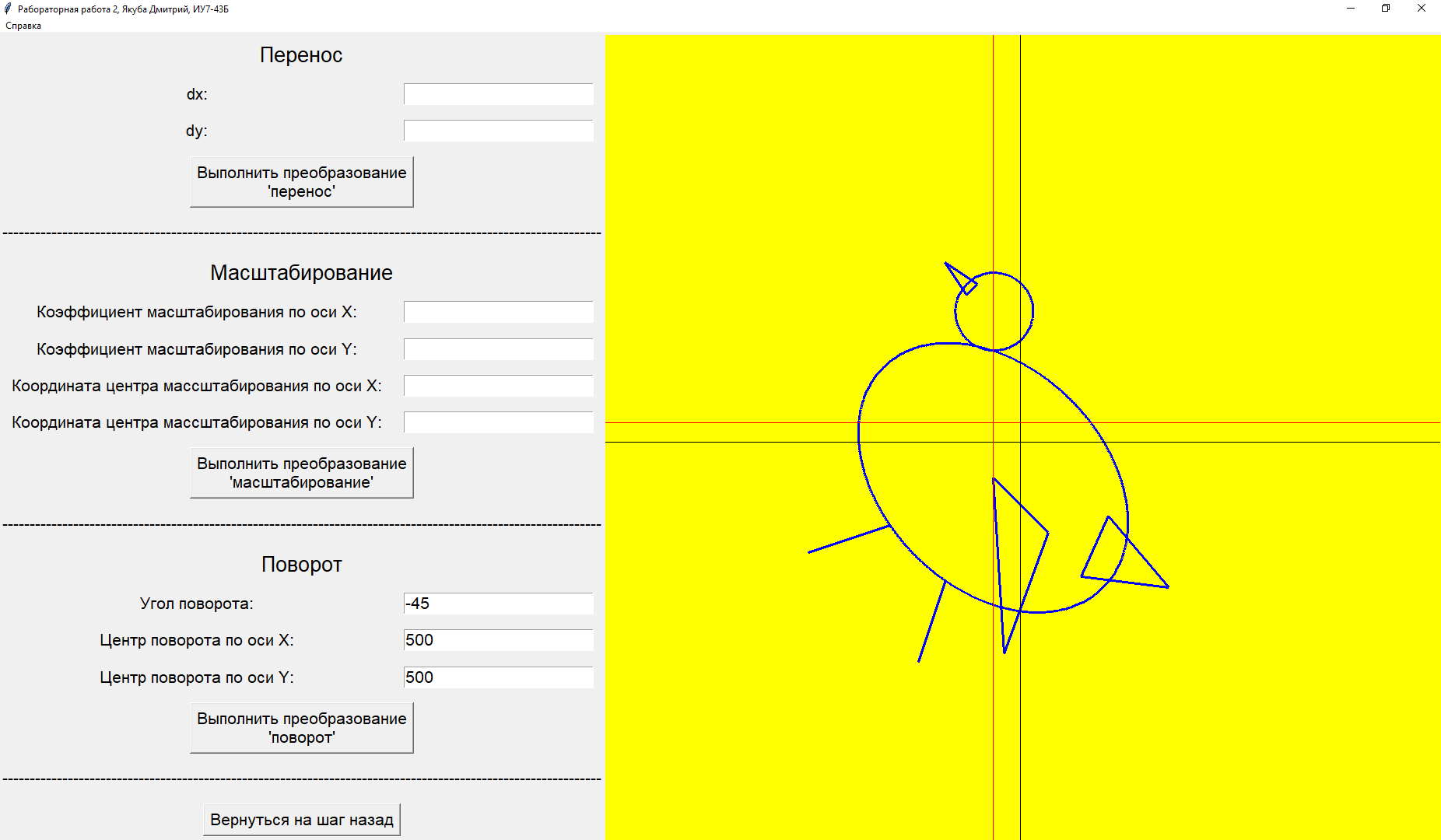
def scaleArray(array, coefX, coefY, centerX, centerY):  
 for i in array:  
 i[0] = i[0] \* coefX + (1 - coefX)\*centerX  
 i[1] = i[1] \* coefY + (1 - coefY)\*centerY  
  
  
def scaleImage(canvasWindow, coefXEnt, coefYEnt, centerXEnt, centerYEnt):  
 try:  
 coefX = coefXEnt.get()  
 coefX = float(coefX)  
 except Exception:  
 makeErrorBadCoefXScale()  
 return  
  
 try:  
 coefY = coefYEnt.get()  
 coefY = float(coefY)  
 except Exception:  
 makeErrorBadCoefYScale()  
 return  
  
 try:  
 centerX = centerXEnt.get()  
 centerX = int(centerX)  
 except Exception:  
 makeErrorBadCenterXScale()  
 return  
  
 try:  
 centerY = centerYEnt.get()  
 centerY = int(centerY)  
 except Exception:  
 makeErrorBadCenterYScale()  
 return  
  
 copyAllToPrev()  
  
 scaleArray(dotsArrayWing, coefX, coefY, centerX, centerY)  
 scaleArray(dotsArrayRightLeg, coefX, coefY, centerX, centerY)  
 scaleArray(dotsArrayEll, coefX, coefY, centerX, centerY)  
 scaleArray(dotsArrayHead, coefX, coefY, centerX, centerY)  
 scaleArray(dotsArrayMouth, coefX, coefY, centerX, centerY)  
 scaleArray(dotsArrayLeftLeg, coefX, coefY, centerX, centerY)  
 scaleArray(dotsArrayTale, coefX, coefY, centerX, centerY)  
  
 printAll(canvasWindow)  
 canvasWindow.create\_line(centerX, 0, centerX, 1055, fill="red")  
 canvasWindow.create\_line(0, centerY, 1075, centerY, fill="red")

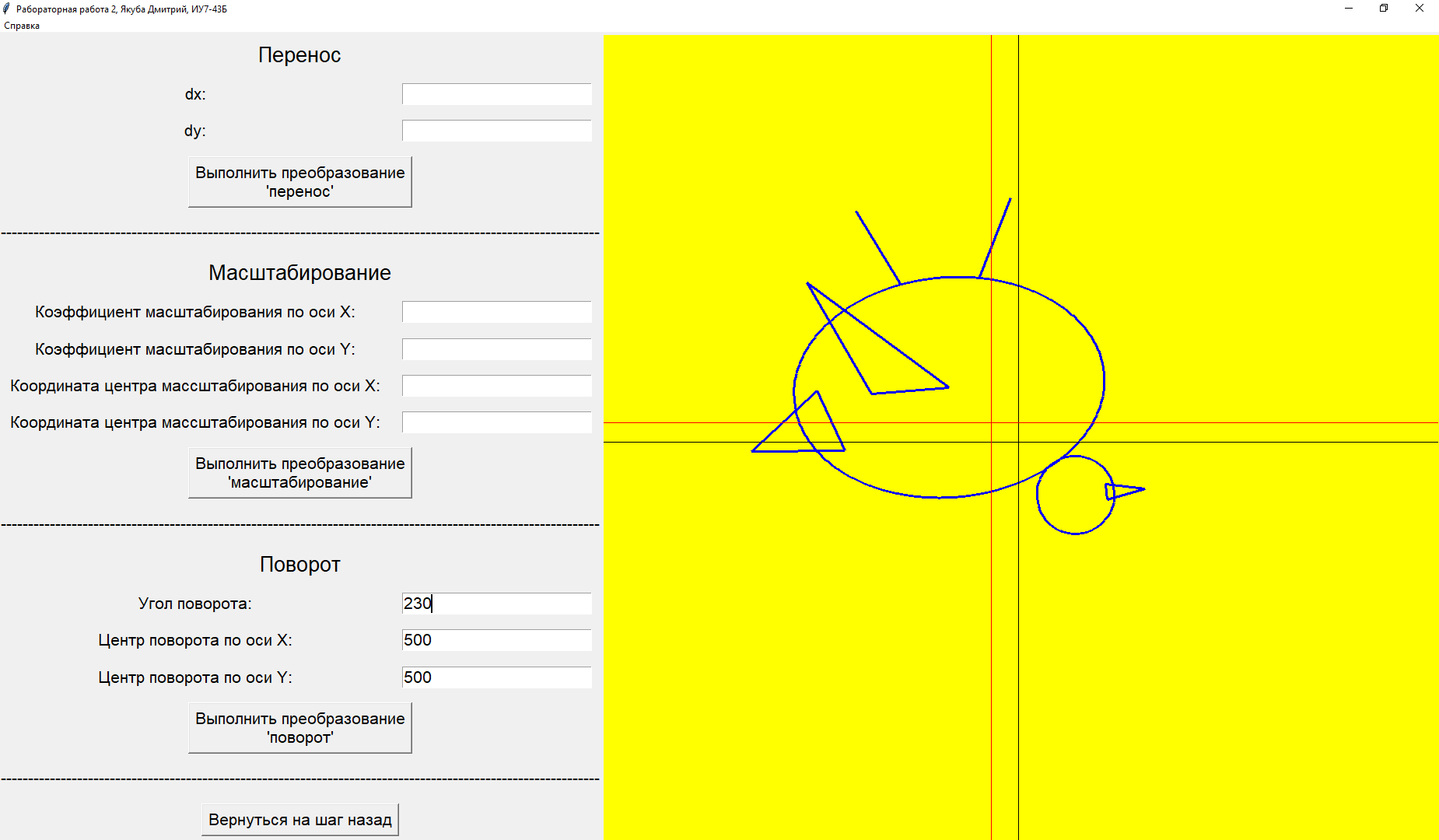
**Операция поворота относительно заданной точки**



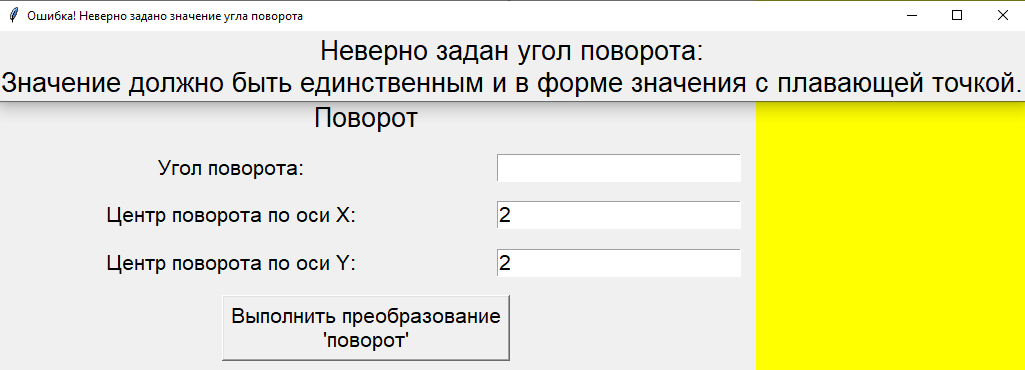




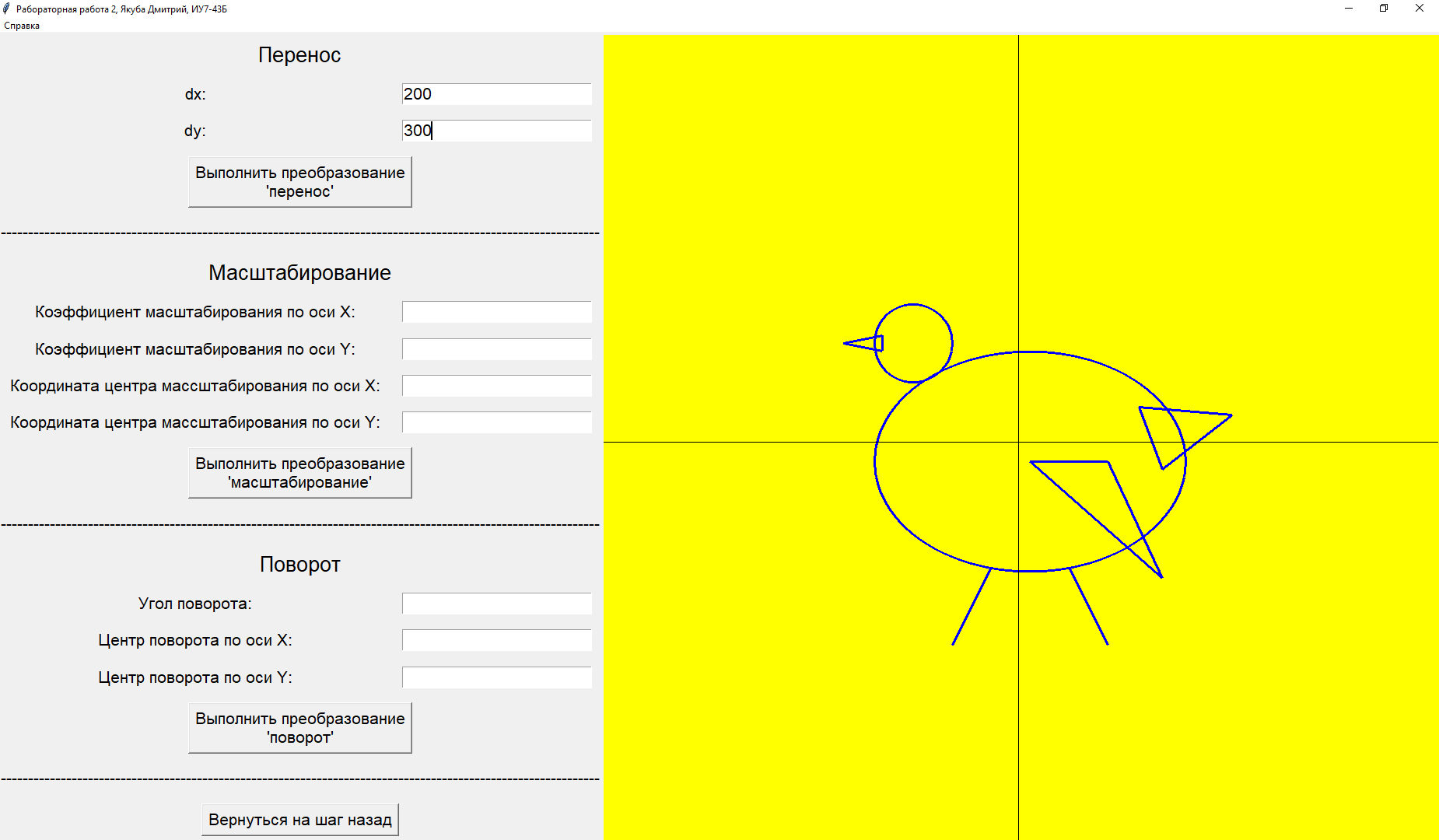


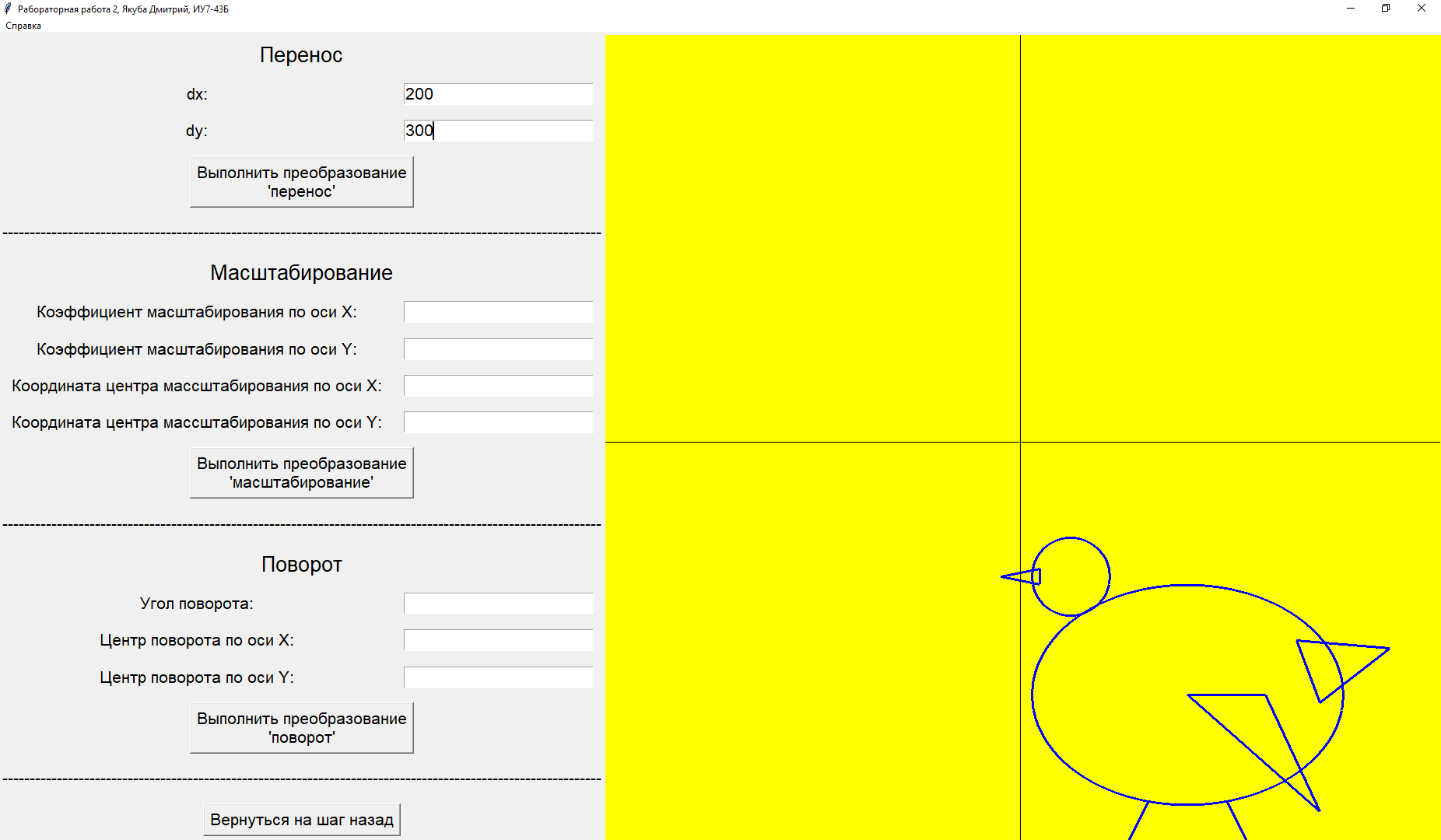


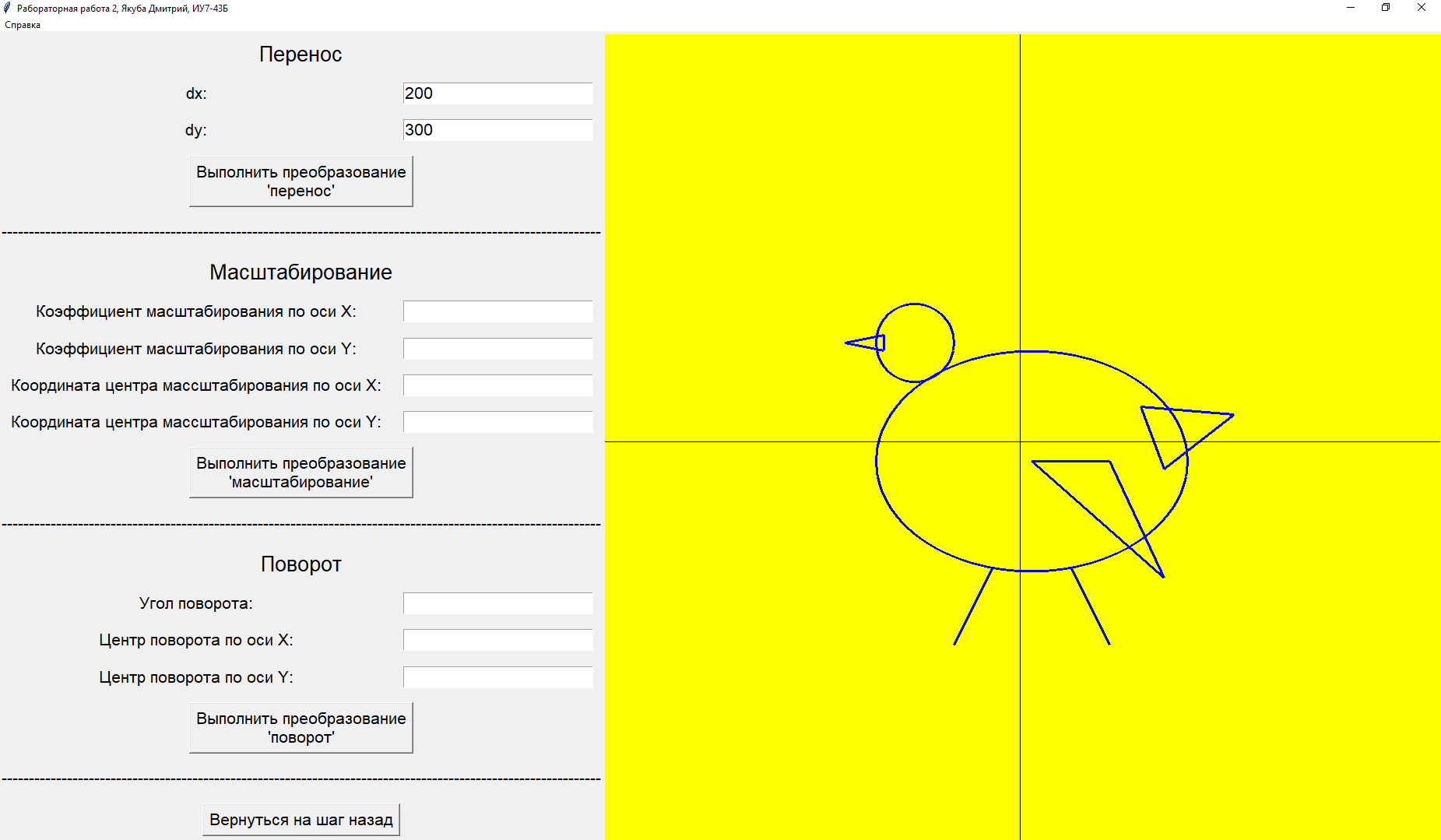
В данной части программы также предусмотрены ошибочные ситуации ввода:

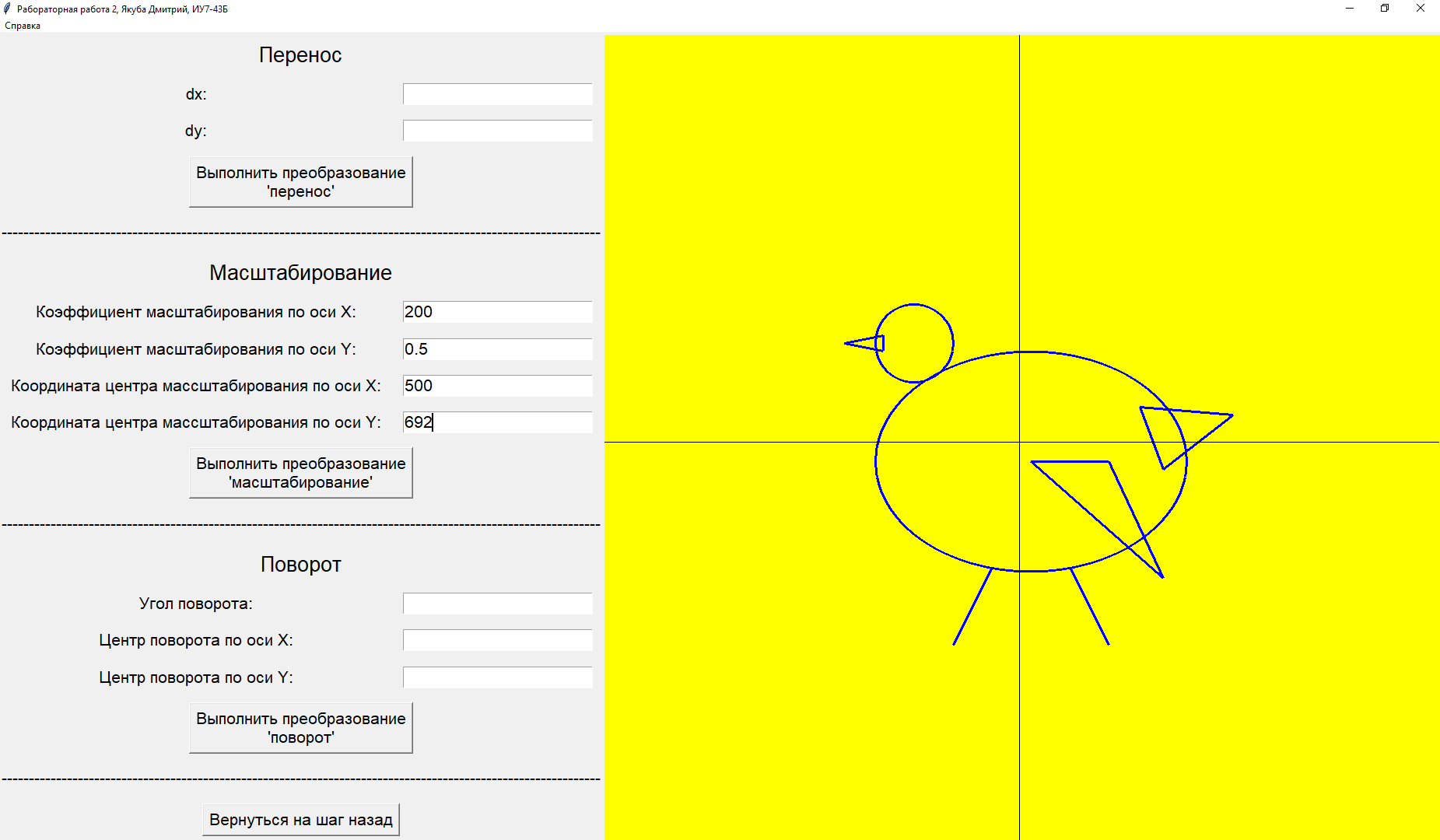


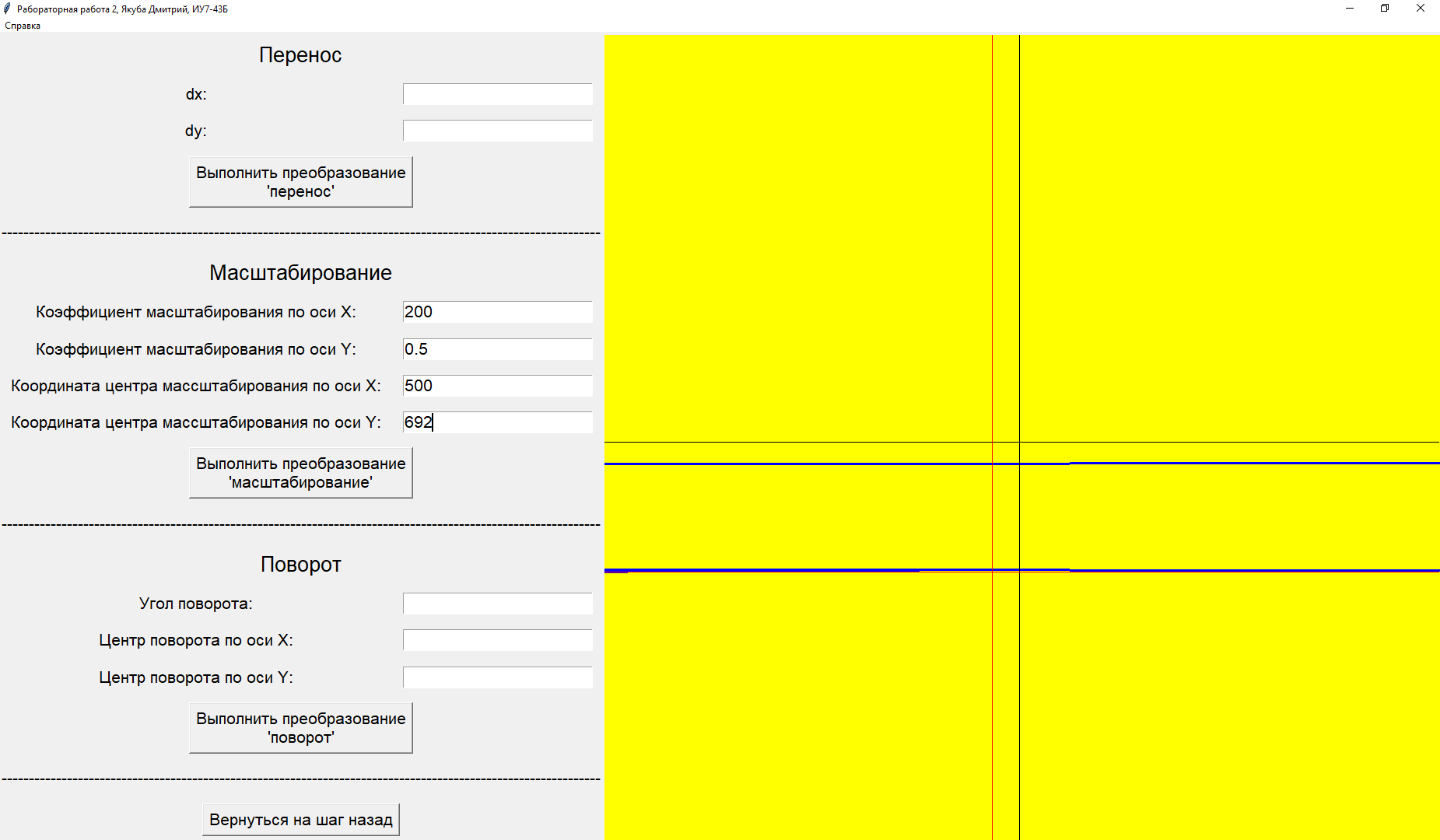
Также проверим возможность возврата в предыдущее состояние рисунка:

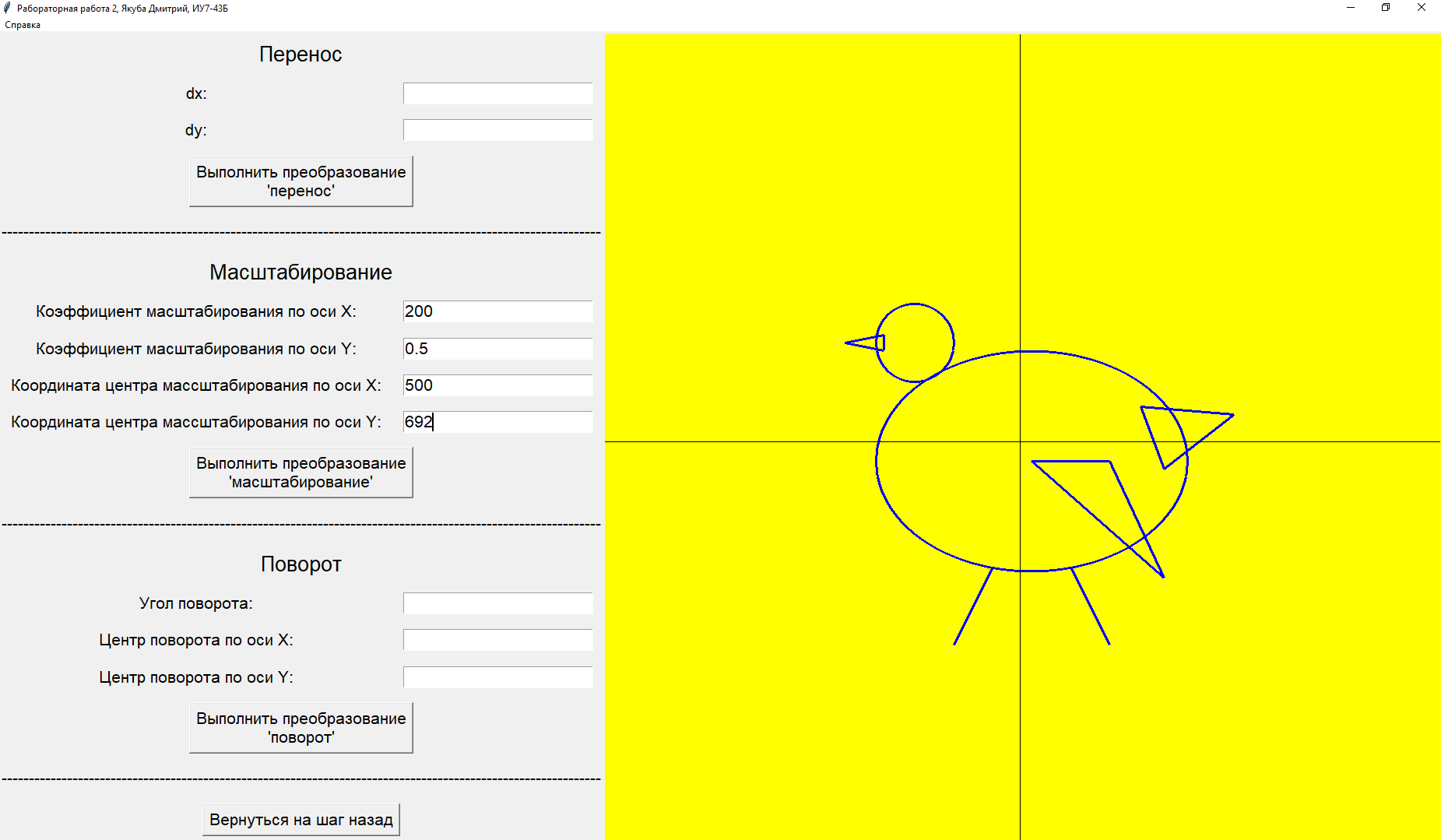


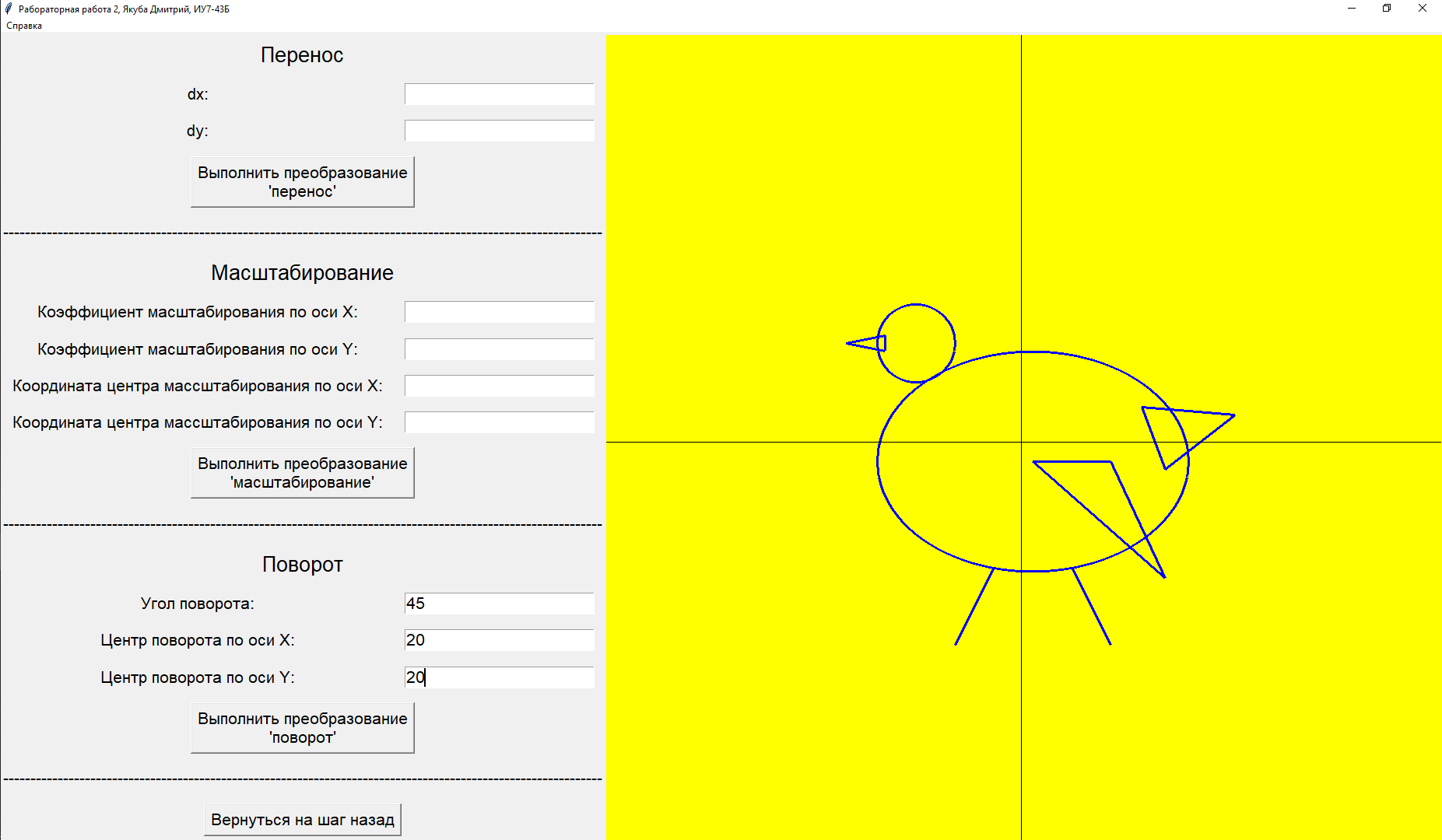


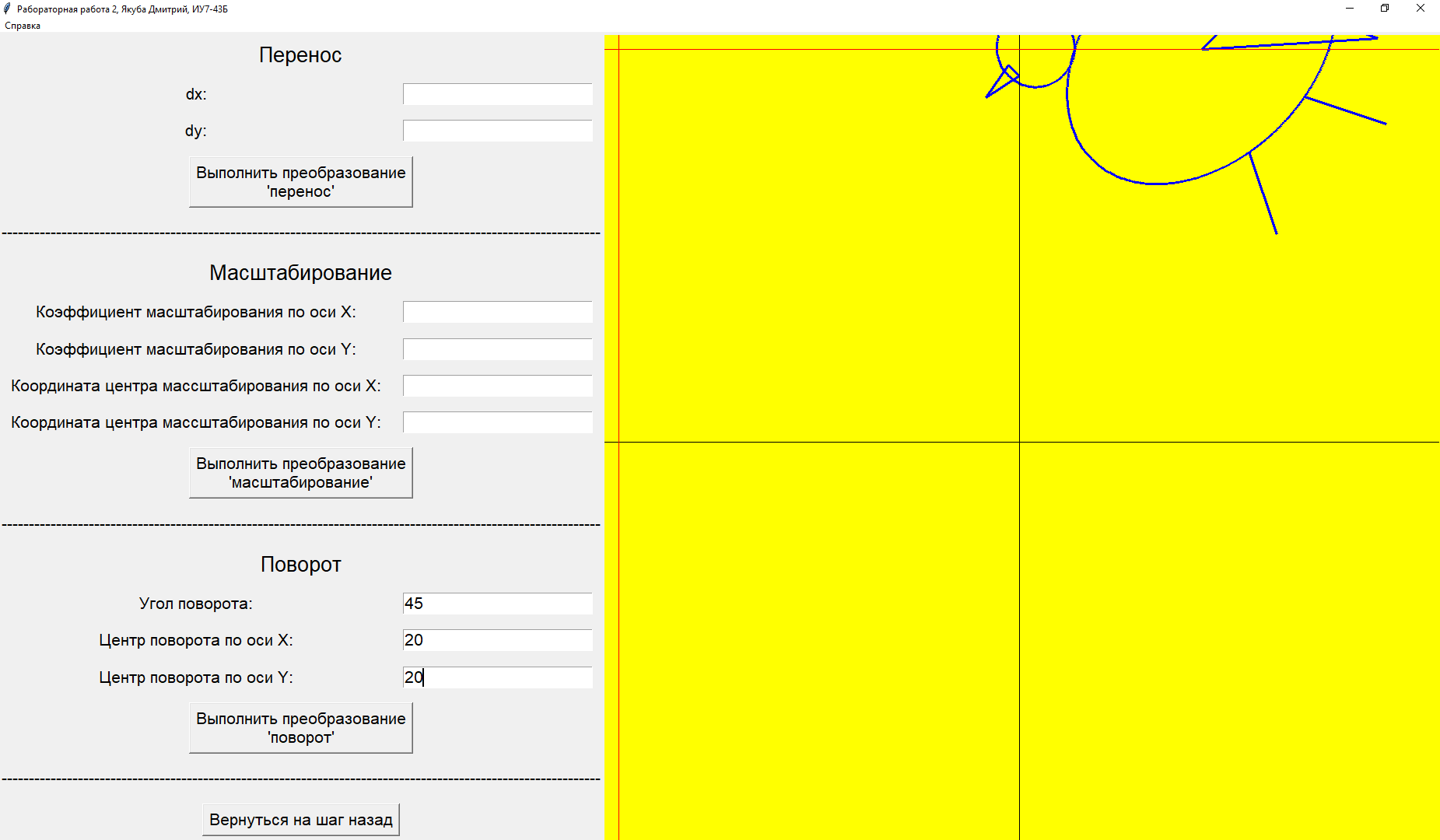


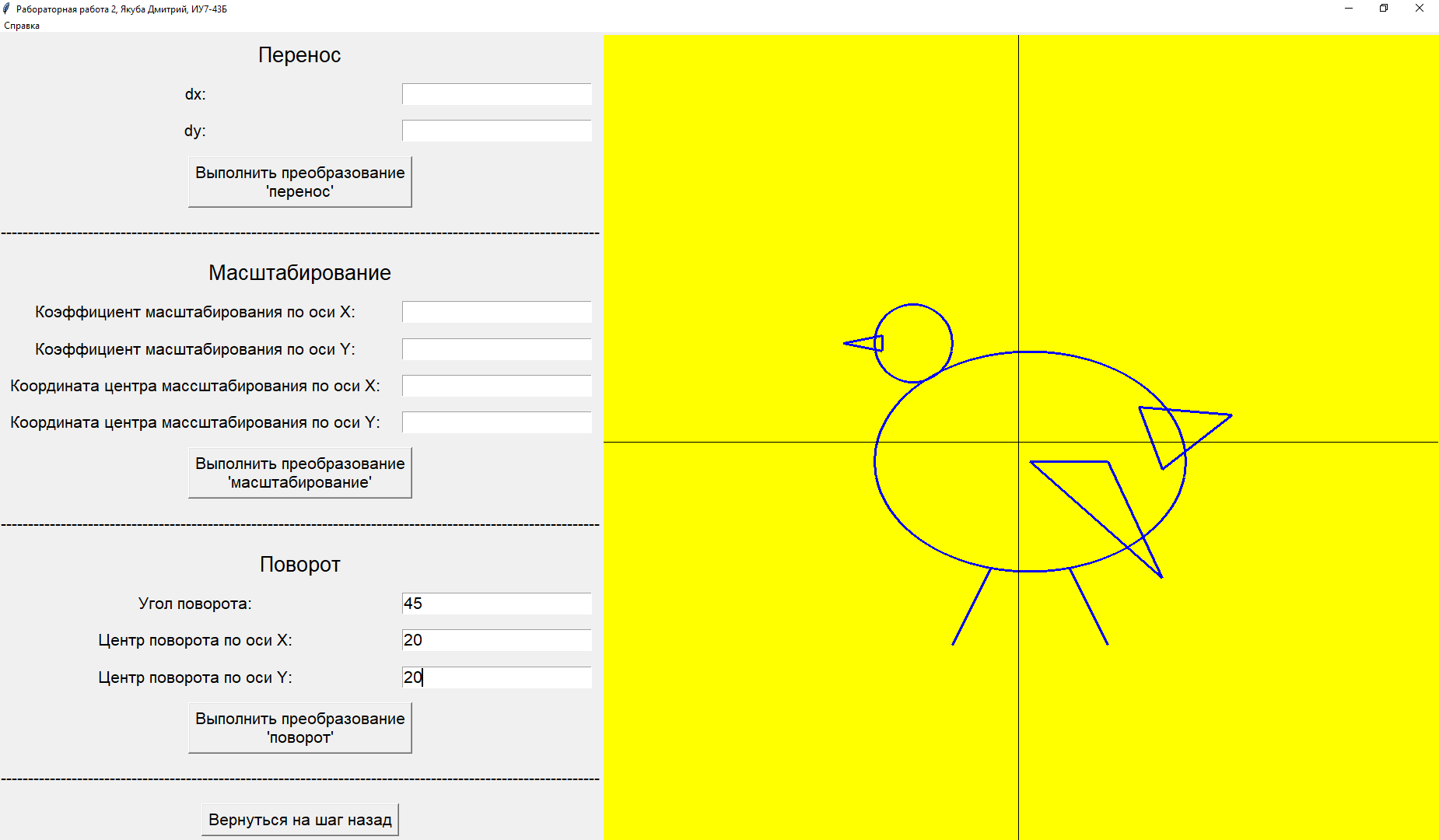












**Алгоритм:**

def turnArray(array, centerX, centerY, cosAngle, sinAngle):  
 for i in array:  
 rememberX = i[0]  
 i[0] = centerX + (i[0] - centerX)\*cosAngle + (i[1] - centerY)\*sinAngle  
 i[1] = centerY + (i[1] - centerY)\*cosAngle - (rememberX - centerX)\*sinAngle  
  
  
def turnImage(canvasWindow, angleEnt, centerXEnt, centerYEnt):  
 try:  
 angle = angleEnt.get()  
 angle = float(angle)  
 except Exception:  
 makeErrorBadAngle()  
 return  
  
 try:  
 centerX = centerXEnt.get()  
 centerX = int(centerX)  
 except Exception:  
 makeErrorBadXTurn()  
 return  
  
 try:  
 centerY = centerYEnt.get()  
 centerY = int(centerY)  
 except Exception:  
 makeErrorBadYTurn()  
 return  
  
 angle = radians(angle)  
 cosAngle = cos(angle)  
 sinAngle = sin(angle)  
  
 copyAllToPrev()  
  
 turnArray(dotsArrayWing, centerX, centerY, cosAngle, sinAngle)  
 turnArray(dotsArrayRightLeg, centerX, centerY, cosAngle, sinAngle)  
 turnArray(dotsArrayEll, centerX, centerY, cosAngle, sinAngle)  
 turnArray(dotsArrayHead, centerX, centerY, cosAngle, sinAngle)  
 turnArray(dotsArrayMouth, centerX, centerY, cosAngle, sinAngle)  
 turnArray(dotsArrayLeftLeg, centerX, centerY, cosAngle, sinAngle)  
 turnArray(dotsArrayTale, centerX, centerY, cosAngle, sinAngle)  
  
 printAll(canvasWindow)  
 canvasWindow.create\_line(centerX, 0, centerX, 1055, fill="red")  
 canvasWindow.create\_line(0, centerY, 1075, centerY, fill="red")

**Механизм хранения предыдущего состояния**

После каждого преобразования изменяемые состояния точек записываются в специальные массивы. Этот способ достаточно требователен по памяти, и вариантом лучше можно было бы предложить сохранять матрицу предыдущего преобразования, что значительно бы уменьшило затраты памяти. Также, с помощью такого способа хранения, можно было бы записывать несколько и из предыдущих преобразований.