Лабораторная работа №4

Графика

Демидова Екатерина Алексеевна

Содержание

# 1 Цель работы

Написать программу на C++ которая визуализирует фигуру и выполняет операции с ней с помощью библиотеки sdl.

# 2 Задание

* Изучить функционал sdl
* Нарисовать фигуру
* Повернуть фигуру с помощзью умножения на матрицу поворота
* Увеличить фигуру

# 3 Выполнение проекта

Для начала мы установили библиотеку sdl и изучили её функционал.

Затем создали класс Engine, который нужен для графической визуализации. В нём написали функцию Render, которая нужна для рендеринга прямоугольника. Ей на вход подаются векторы, по которым он строится. В этой функции используются встроенные в библиотеку sdl функции, такие как SDL\_SetRenderDrawColor() - задаёт цвет, SDL\_RenderDrawLine() - рисует линии, SDL\_RenderPresent() - обновляет экран после рендеринга.

void Render(Vector &v1, Vector &v2, Vector &v3, Vector &v4)  
 {  
  
  
   
 SDL\_SetRenderDrawColor(renderer, 0, 0, 0, SDL\_ALPHA\_OPAQUE);  
 SDL\_RenderClear(renderer);  
 SDL\_SetRenderDrawColor(renderer, 255, 255, 255, SDL\_ALPHA\_OPAQUE);  
  
 SDL\_RenderDrawLine(renderer, v1[0], v1[1], v2[0], v2[1]);  
 SDL\_RenderDrawLine(renderer, v2[0], v2[1], v3[0], v3[1]);  
 SDL\_RenderDrawLine(renderer, v3[0], v3[1], v4[0], v4[1]);  
 SDL\_RenderDrawLine(renderer, v4[0], v4[1], v1[0], v1[1]);  
 SDL\_RenderPresent(renderer);  
 };

Также была реализована функция для отрисовки фигуры, ей также на вход подаются векторы задающие координаты углов прямоугольник:

void Run(Vector &vec1, Vector &vec2, Vector &vec3, Vector &vec4)  
 {  
 Init();  
 while (!flagToExit)  
 {  
 PollsEvent();  
 Update();  
 Render(vec1, vec2, vec3, vec4);  
 }  
 SDL\_DestroyRenderer(renderer);  
 std::cout << "renderer memory finalized"  
 << "\n";  
 SDL\_DestroyWindow(window);  
 std::cout << "window memory finalized"  
 << "\n";  
 SDL\_Quit();  
 std::cout << "SDL memory finalized"  
 << "\n";  
 }

Перейдём к функции main. В ней был реализован switch-case, в котором пользователю даётся возможность просто нарисовать фигуру, повернуть её и увеличить/уменьшить. Эти возможности реализованы с помощью умножения матриц на вектора.

case 1:  
 {  
 float grad;  
 cout << "На какой угол хотите повернуть? От 0 до 90!!!!\n";  
   
 cin >> grad;  
 Matrix matr(2,2);  
 matr[0][0] = cos(3.1416\*grad/180);  
 matr[0][1] = sin(3.1416\*grad/180);  
 matr[1][0] = -sin(3.1416\*grad/180);  
 matr[1][1] = cos(3.1416\*grad/180);  
   
 Vector v1 = matr\*vect1;  
 Vector v2 = matr\*vect2;  
 Vector v3 = matr\*vect3;  
 Vector v4 = matr\*vect4;  
   
 Engine visualMath = Engine(1000, 1000);  
 visualMath.Run(v1,v2,v3,v4);  
 break;  
 }

case 2:  
 {  
 float grad;  
 cout << "Во сколько хотите увеличить фигуру? От 0 до 2\n";  
 cin >> grad;  
 Matrix matr(2,2);  
 matr[0][0] = grad;  
 matr[0][1] = 0;  
 matr[1][0] = 0;  
 matr[1][1] = grad;  
   
 Vector v1 = matr\*vect1;  
 Vector v2 = matr\*vect2;  
 Vector v3 = matr\*vect3;  
 Vector v4 = matr\*vect4;  
   
 Engine visualMath = Engine(1000, 1000);  
 visualMath.Run(v1,v2,v3,v4);  
 break;  
 }

case 3:  
 Engine visualMath = Engine(1000, 1000);  
 visualMath.Run(vect1,vect2,vect3,vect4);

Приведём пример использования этих функций. (рис. 1-4)

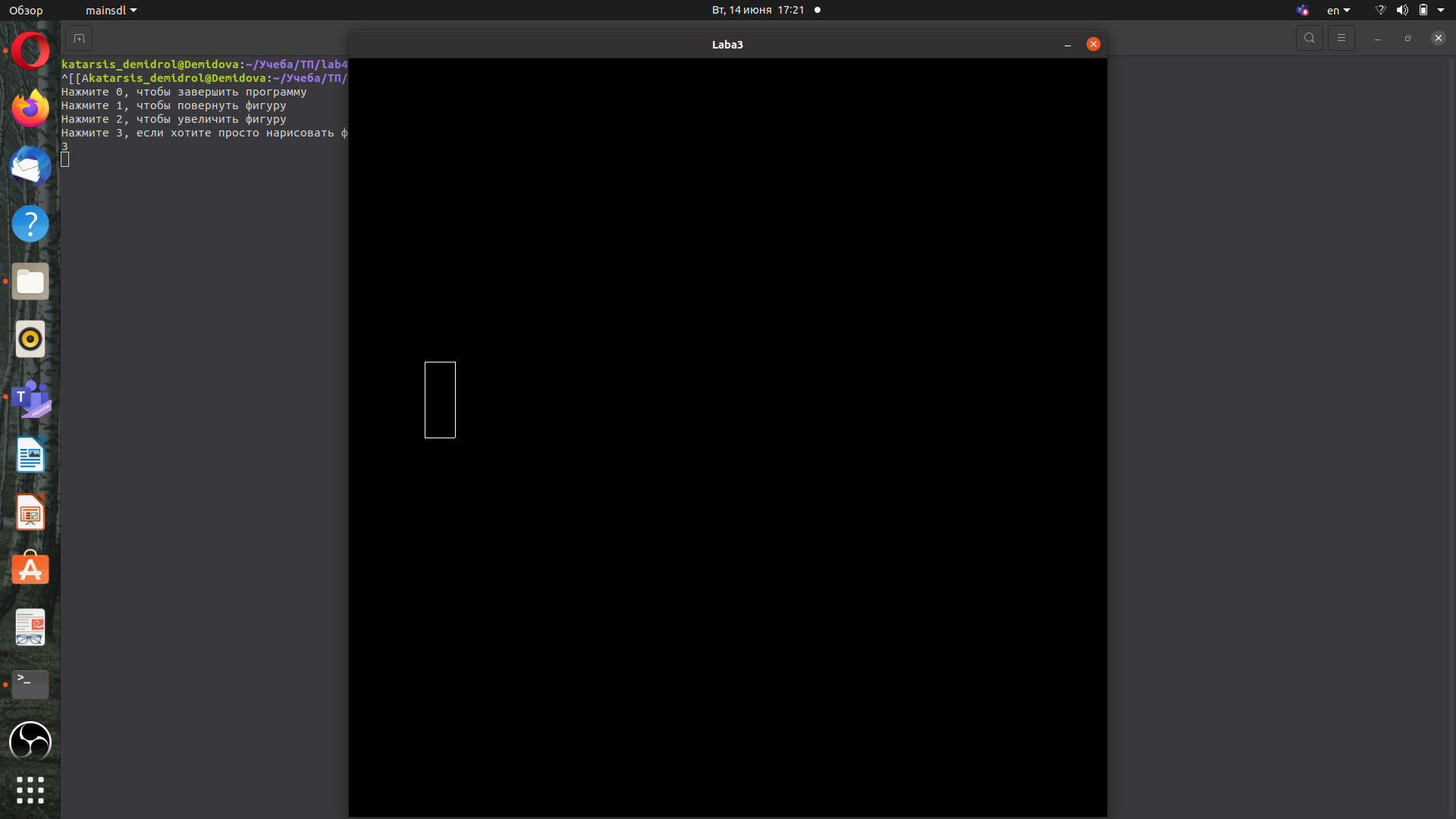


Рис. 1: Просто отрисовка

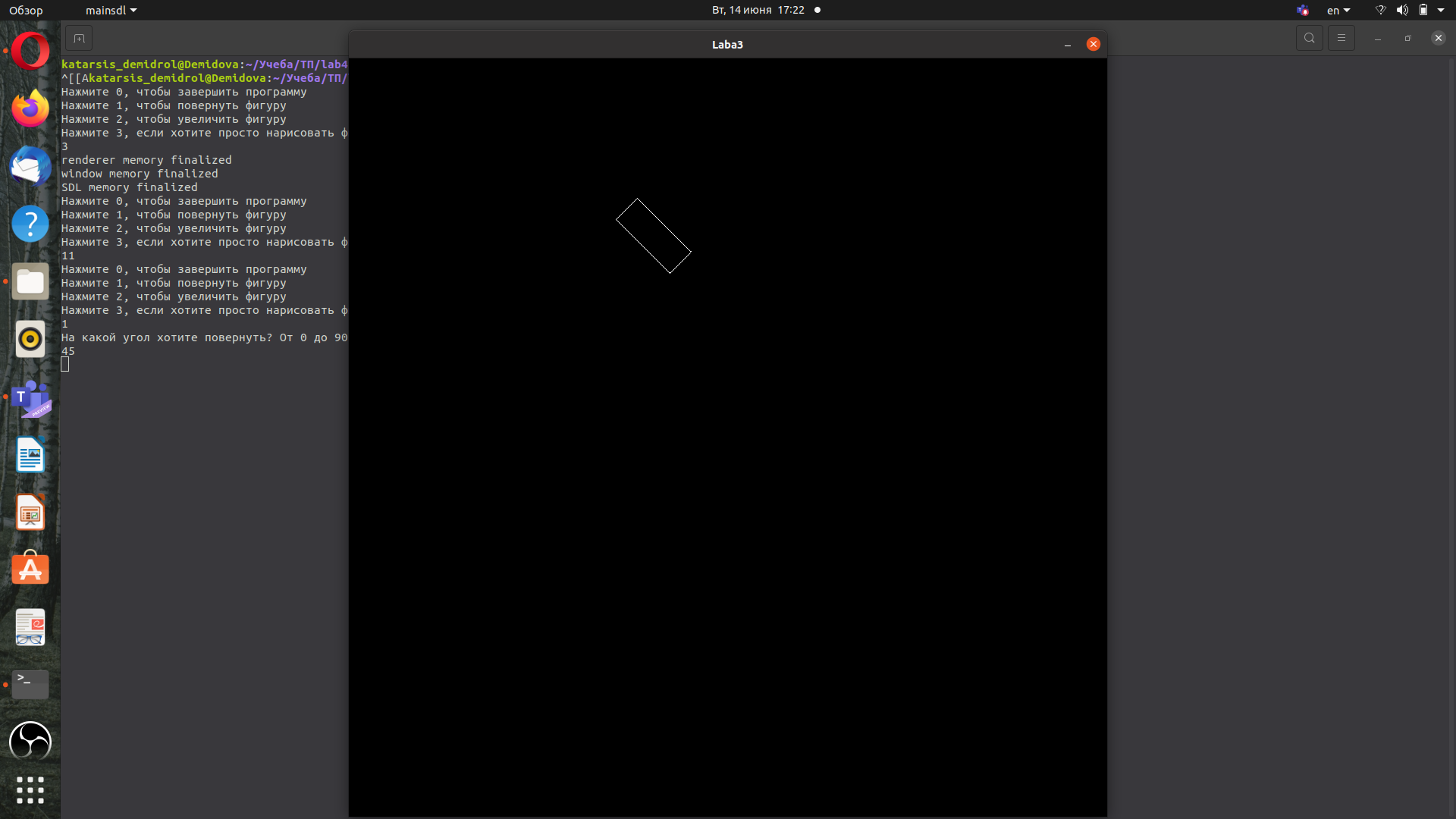


Рис. 2: Поворот

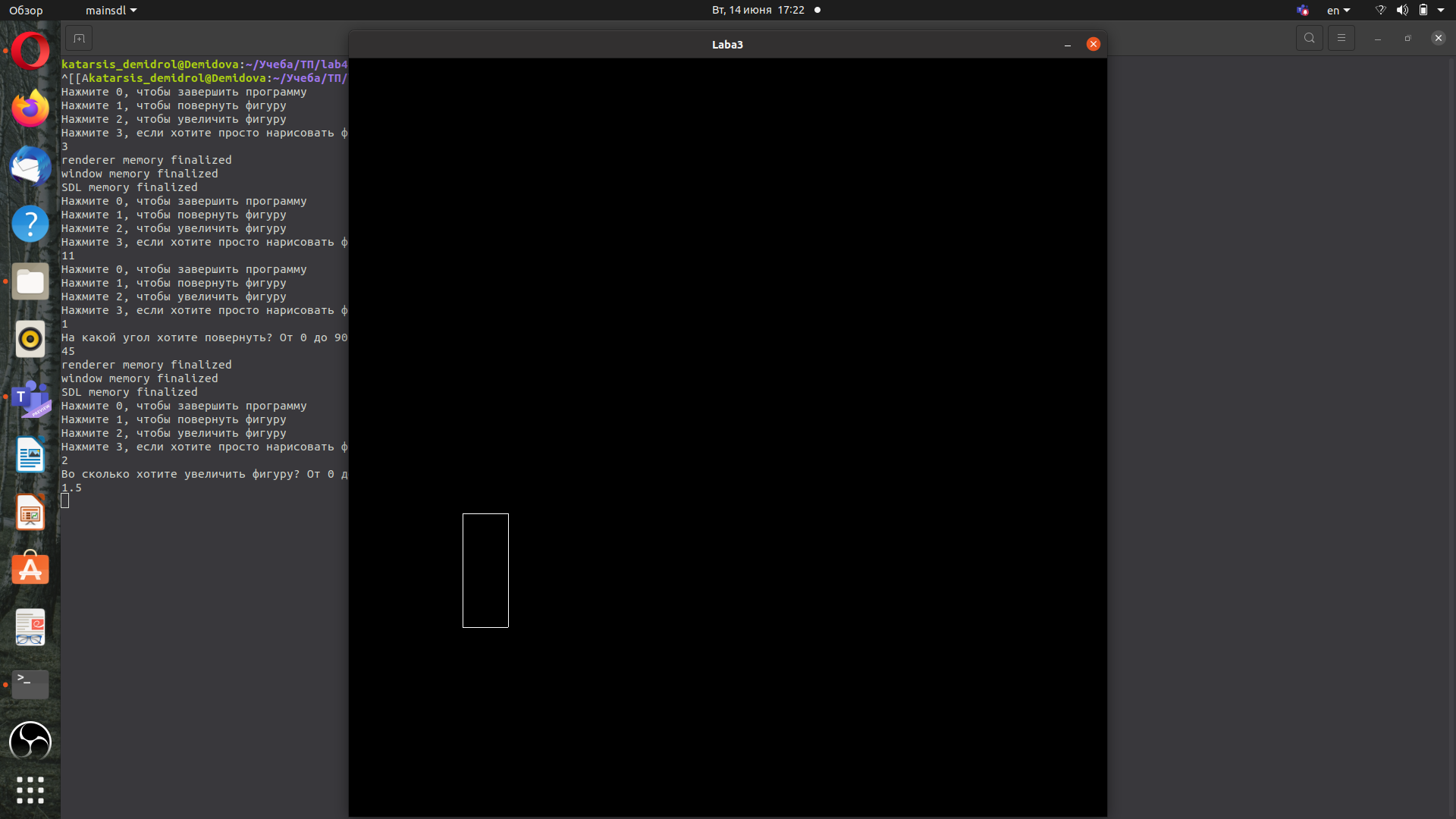


Рис. 3: Масштабирование. Увеличение

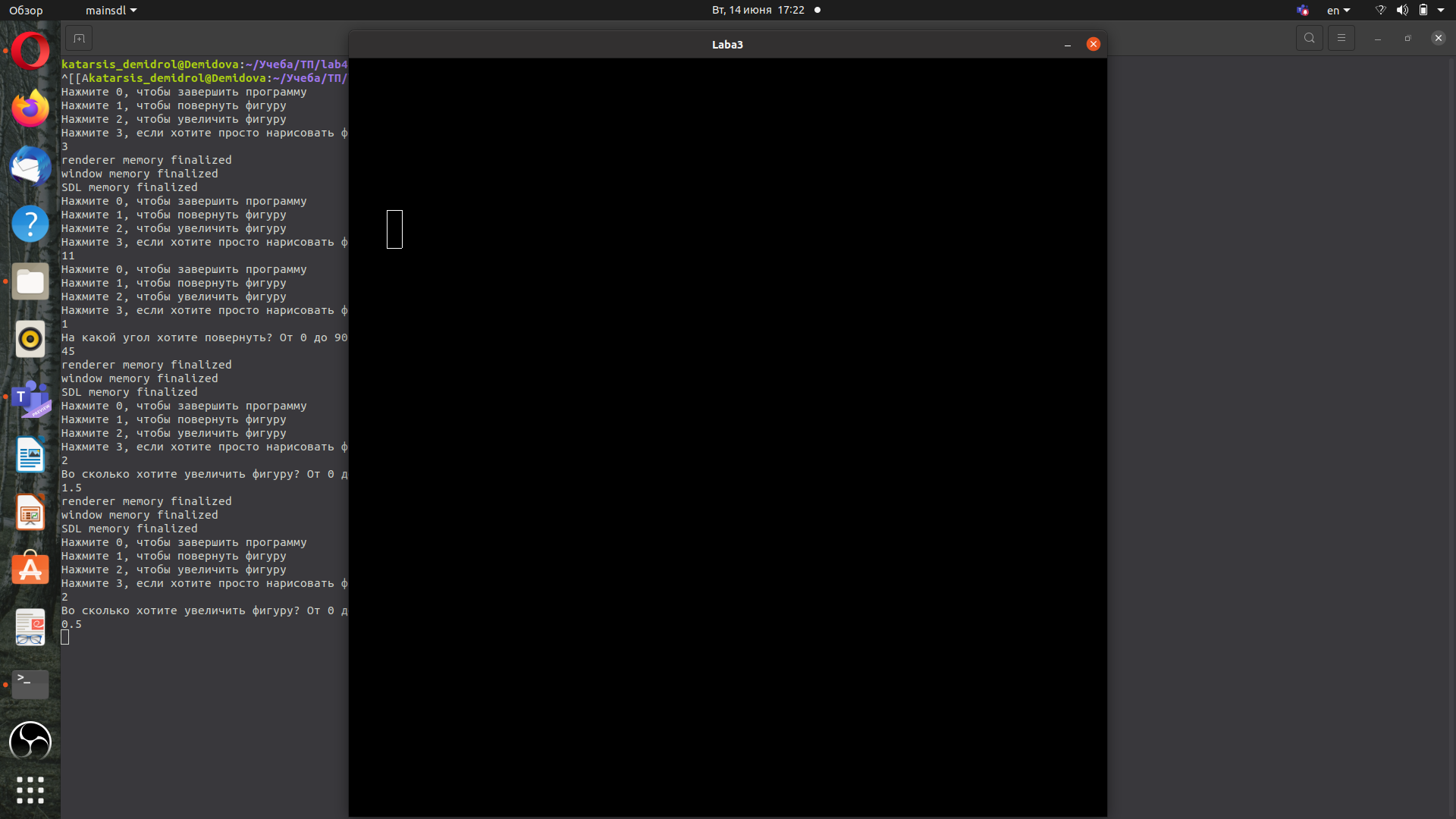


Рис. 4: Масштабирование. Уменьшение

# 4 Выводы

В результате выполнения лабораторной работы были получены практические навыки работы с библиотекой sdl, была написана программа на языке C++, в которой осуществляется графическое отображение фигуры и операции над ней с помощью матриц.