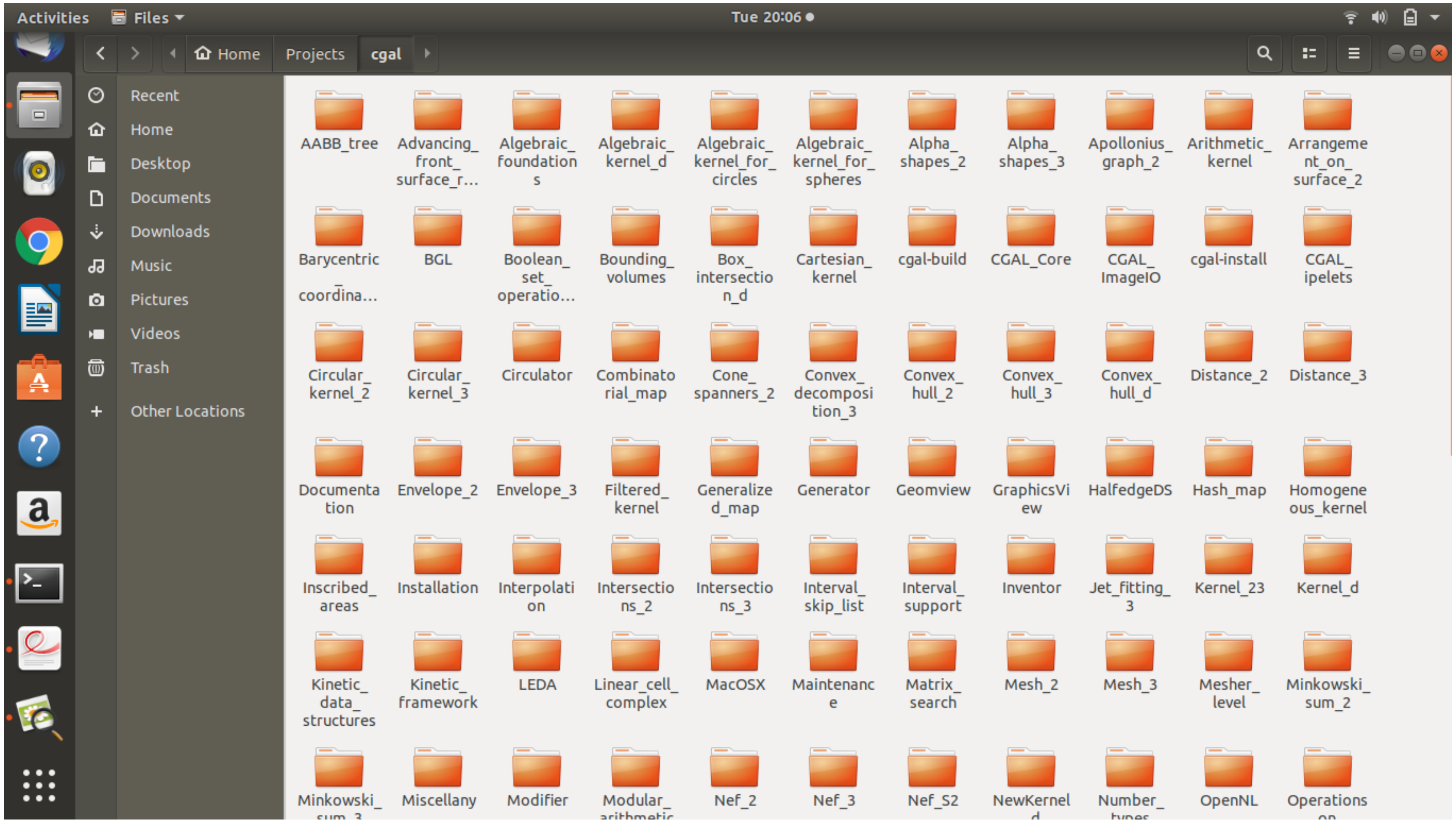
CGAL Demo

- In this presentation, we will see
  - ➢ How to run an already existing demo in CGAL – GraphicsView
  - ➢ How to modify it to create your own demo
  - ➢ How to do a sample program

1. How to run an already existing demo in CGAL – GraphicsView

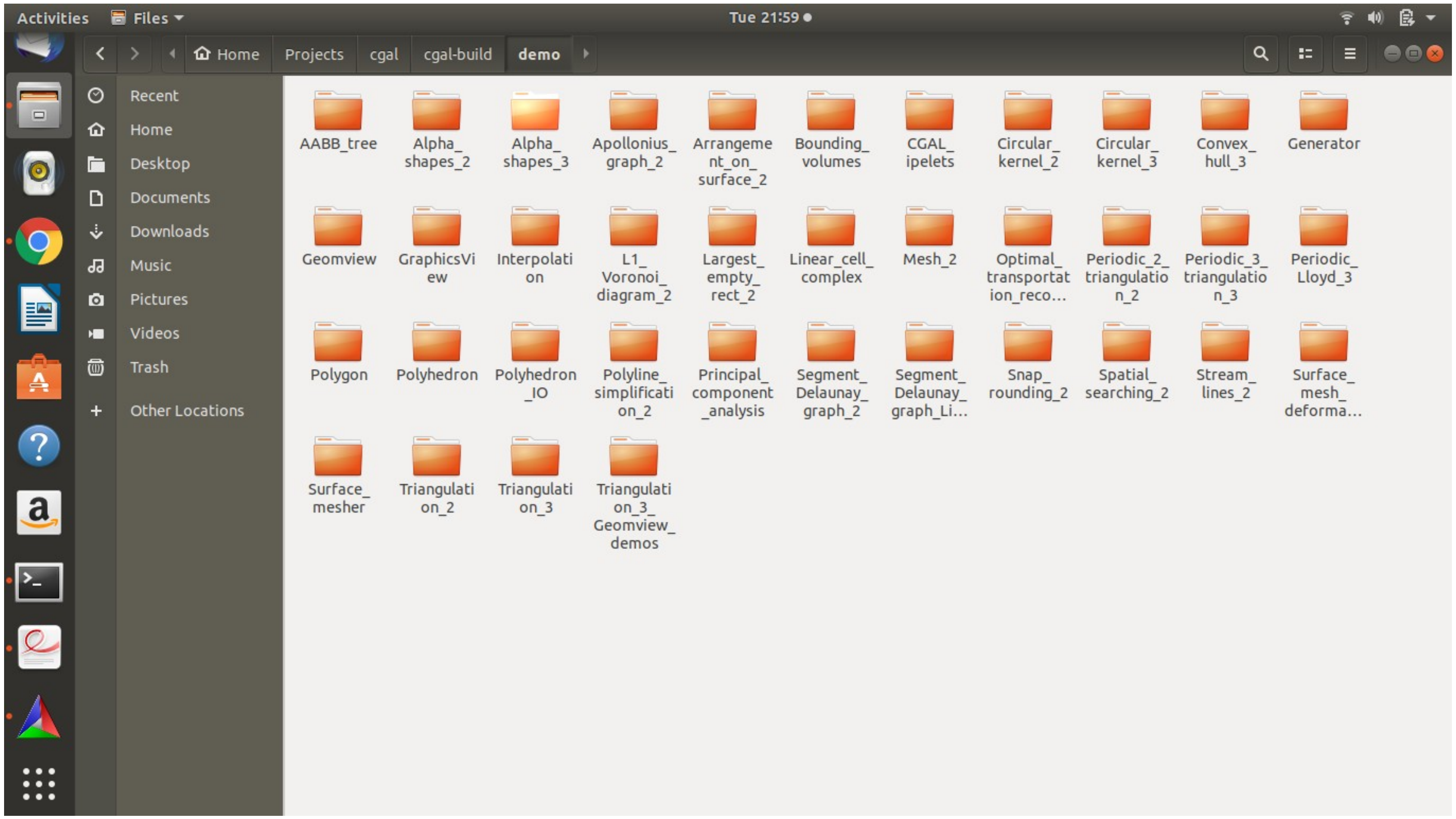
Run demo GraphicsView

- Steps
  1. Open your CGAL folder



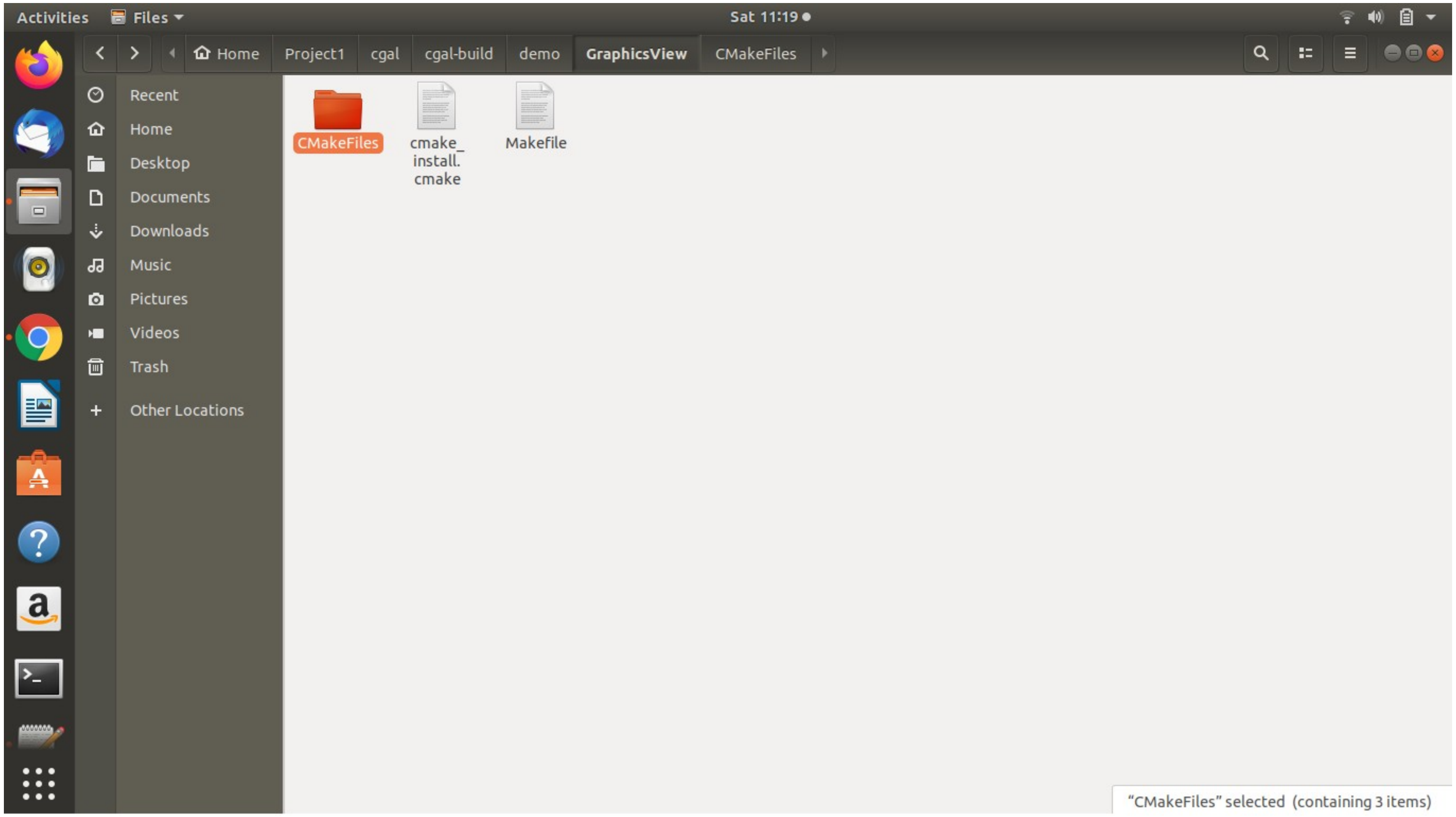
Run demo GraphicsView

- Steps
  2. Go to cgal/build/demo



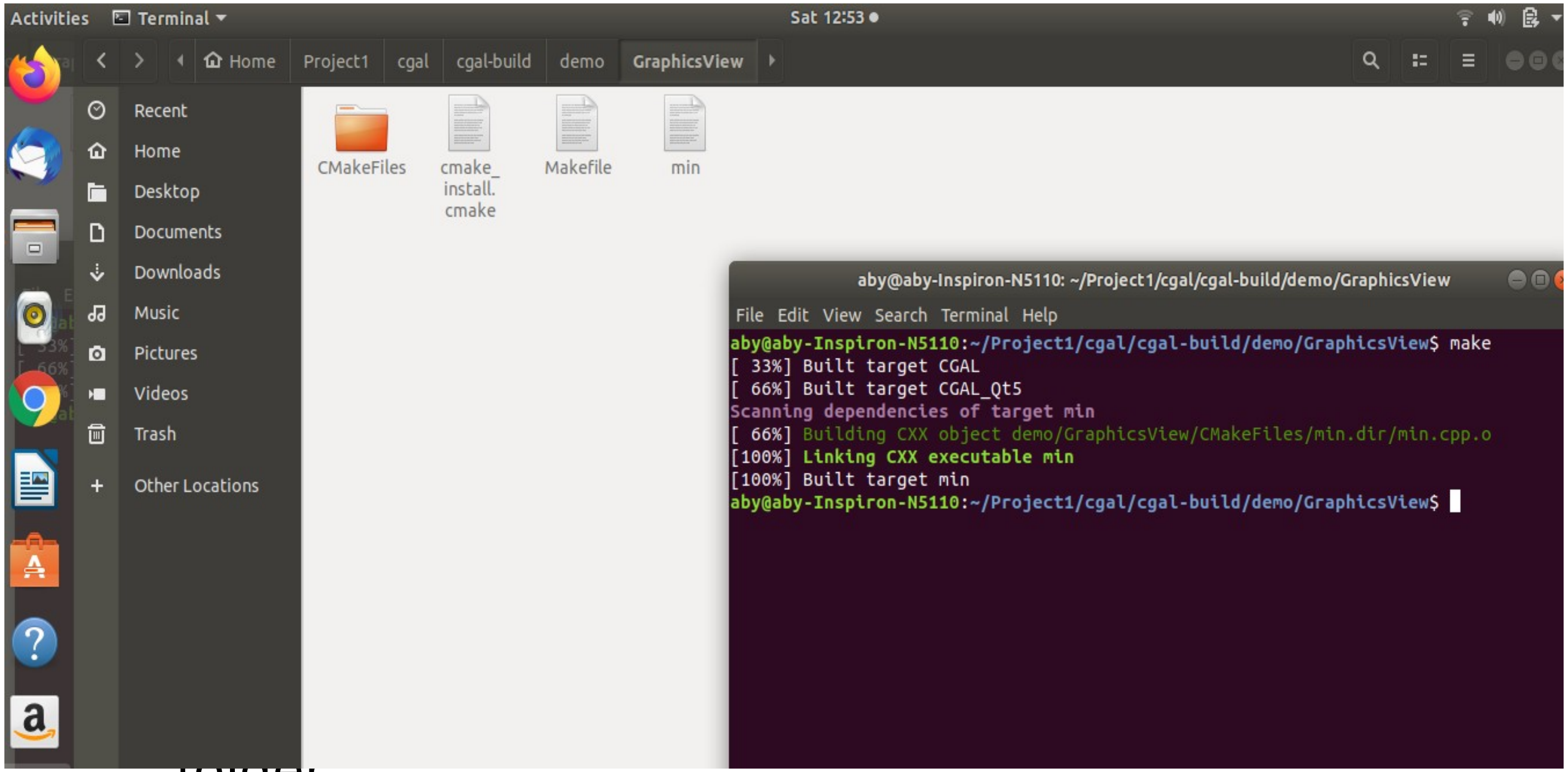
Run demo GraphicsView

- Steps
  3. Open folder GraphicsView



Run demo GraphicsView

- Steps
  4. Open terminal inside this folder and create the executable by **`make`** command

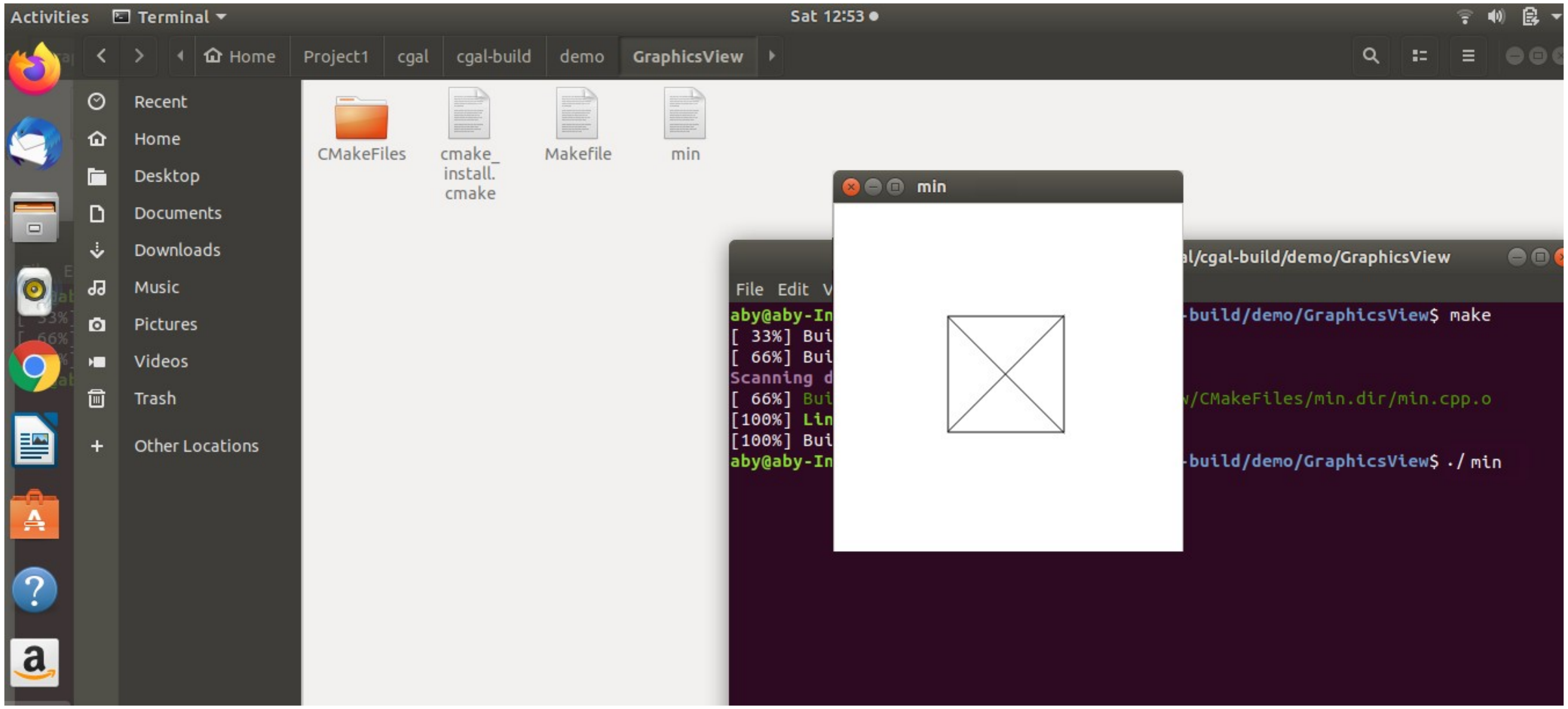


```
aby@aby-Inspiron-N5110:~/Project1/cgal/cgal-build/demo/GraphicsView$ make
[ 33%] Built target CGAL
[ 66%] Built target CGAL_Qt5
Scanning dependencies of target min
[ 66%] Building CXX object demo/GraphicsView/CMakeFiles/min.dir/min.cpp.o
[100%] Linking CXX executable min
[100%] Built target min
aby@aby-Inspiron-N5110:~/Project1/cgal/cgal-build/demo/GraphicsView$
```

folder

Run demo GraphicsView

- Steps
  5. Run the executable either by clicking it or by typing `./min` in the terminal



```
al/cgal-build/demo/GraphicsView
aby@aby-In                    -build/demo/GraphicsView$ make
[ 33%] Bui
[ 66%] Bui
Scanning d
[ 66%] Bui                    /CMakeFiles/min.dir/min.cpp.o
[100%] Lin
[100%] Bui
aby@aby-In                    -build/demo/GraphicsView$ ./min
```

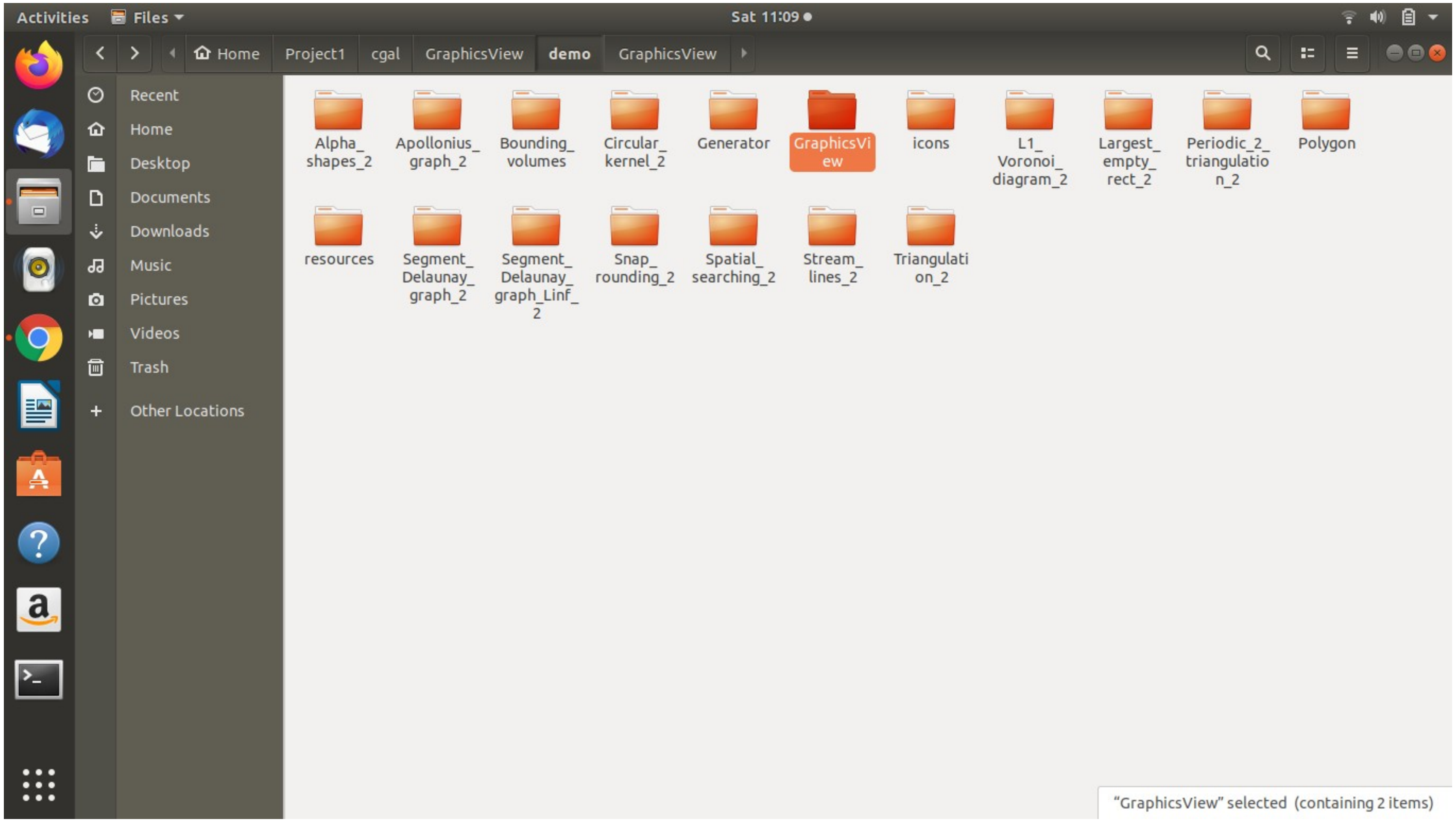
-This demo draws a rectangle and two diagonals of it in a QT window.

Code of min.cpp in GraphicsView

- Let us see the code of this demo.
- Source Code of all the demos can be found in the CGAL folder.
- Open the folder CGAL/demo/GraphicsView

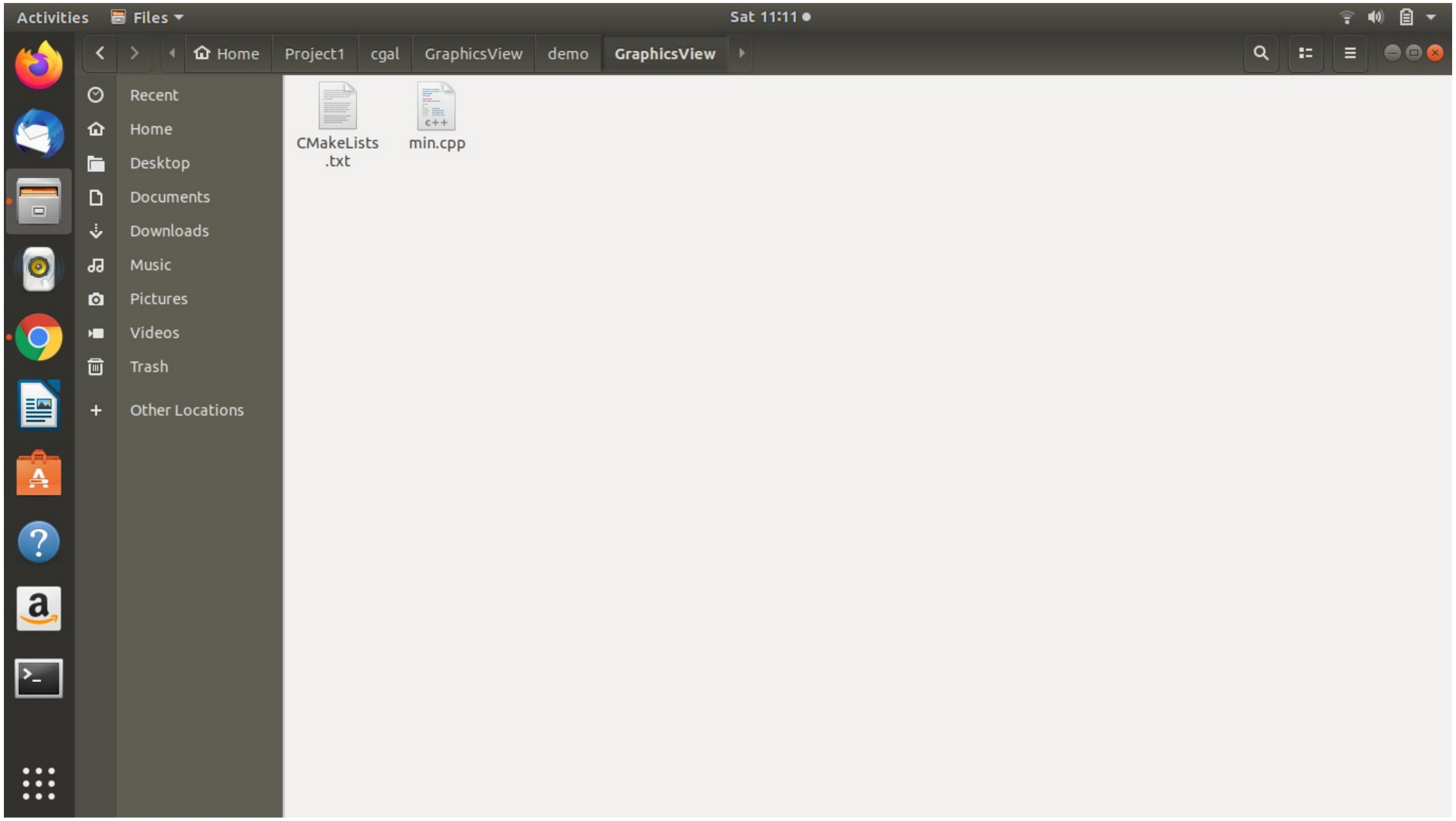
Code of min.cpp in GraphicsVie

- Open the folder GraphicsView



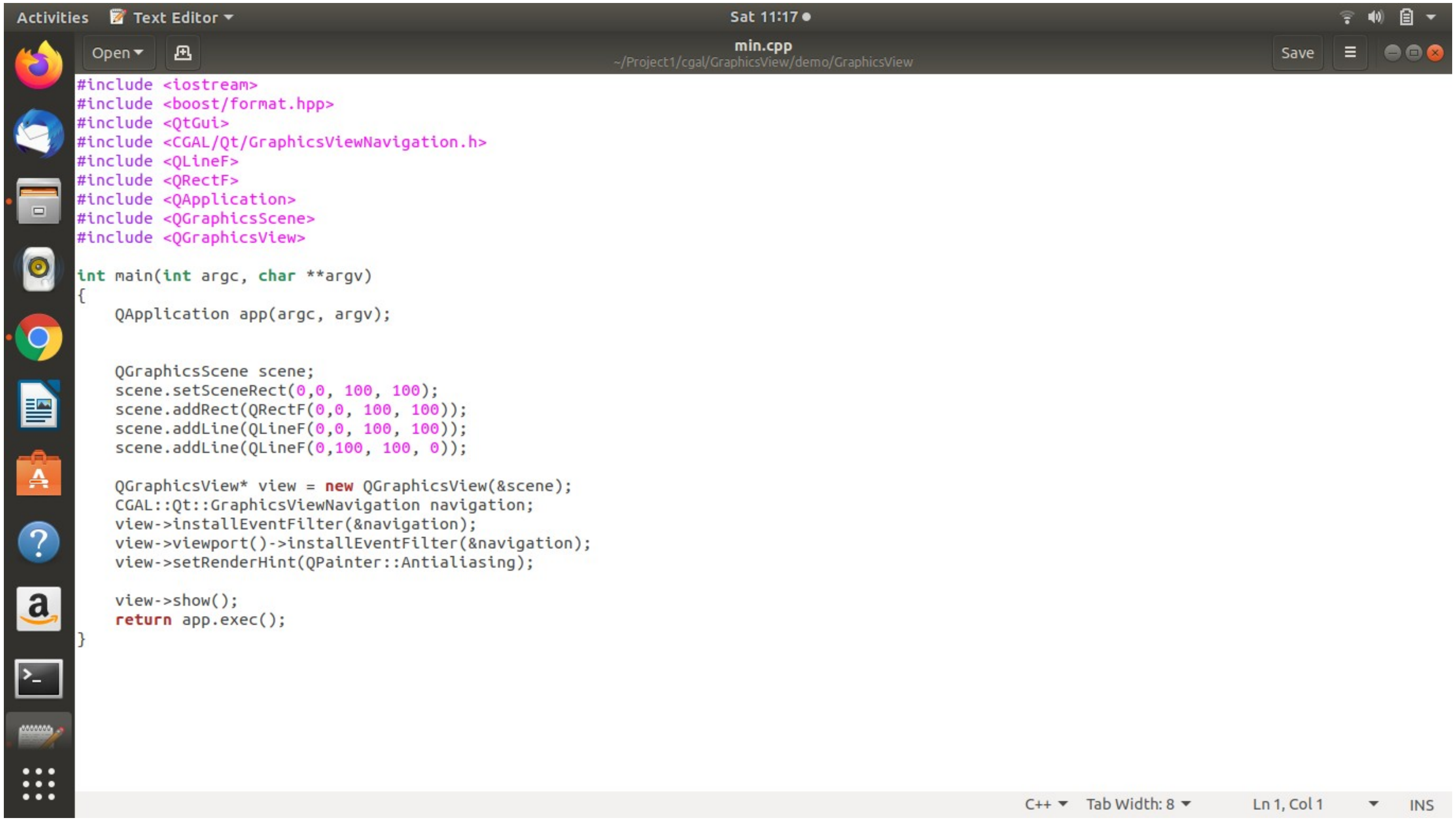
Code of min.cpp in GraphicsVie

- There are two files. One .txt file and one .cpp file



Code of min.cpp in GraphicsView

- Open min.cpp



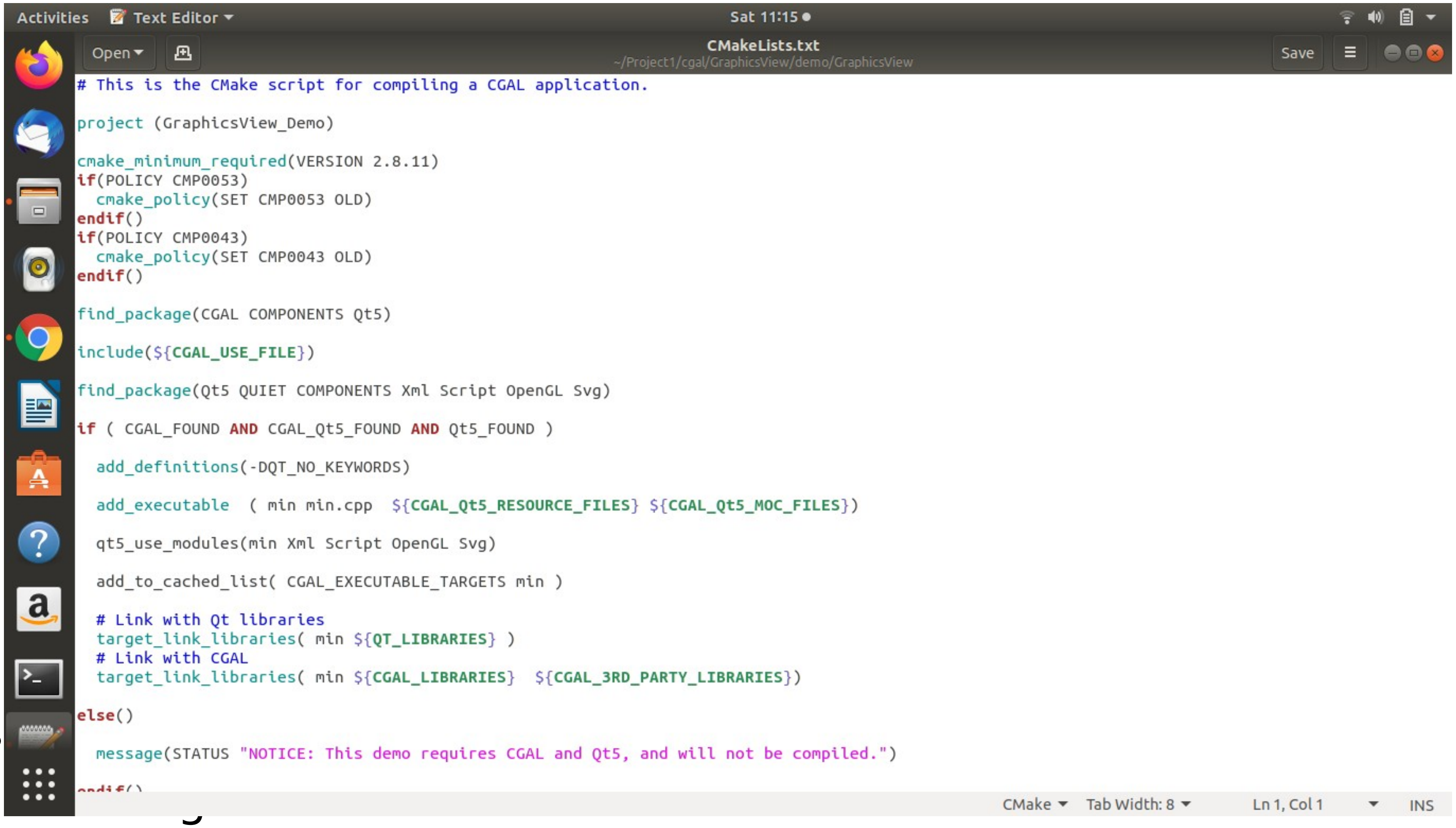
```cpp
#include <iostream>
#include <boost/format.hpp>
#include <QtGui>
#include <CGAL/Qt/GraphicsViewNavigation.h>
#include <QLineF>
#include <QRectF>
#include <QApplication>
#include <QGraphicsScene>
#include <QGraphicsView>

int main(int argc, char **argv)
{
    QApplication app(argc, argv);


    QGraphicsScene scene;
    scene.setSceneRect(0,0, 100, 100);
    scene.addRect(QRectF(0,0, 100, 100));
    scene.addLine(QLineF(0,0, 100, 100));
    scene.addLine(QLineF(0,100, 100, 0));

    QGraphicsView* view = new QGraphicsView(&scene);
    CGAL::Qt::GraphicsViewNavigation navigation;
    view->installEventFilter(&navigation);
    view->viewport()->installEventFilter(&navigation);
    view->setRenderHint(QPainter::Antialiasing);

    view->show();
    return app.exec();
}
```

Min.cpp

```
#include <iostream>  // cpp header
#include <boost/format.hpp> //boost header (not compulsory)
#include <QtGui>     // QT header for GUI
#include <CGAL/Qt/GraphicsViewNavigation.h>  //CGAL header for linking CGAL and
   QGraphicsView
#include <QLineF>  // QT header for Line functions
#include <QRectF>  // QT header for Rectangle functions
#include <QApplication> // QT header for creating a QT GUI application
#include <QGraphicsScene> // QT header for 2D graphical items
#include <QGraphicsView> // QT header for displaying the contents of a scene

int main(int argc, char **argv)
{
    QApplication app(argc, argv);  //create a new QT GUI application

    QGraphicsScene scene;         //create a QT scene to display point
    scene.setSceneRect(0,0, 100, 100);  //set scene size
    scene.addRect(QRectF(0,0,100,100);  //add a rectangle in the scene
    scene.addLine(QLineF(0,0,100,100); //add a line in the scene
    scene.addLine(QLineF(0,100,100,0); //add a line in the scene

   //display the scene
    QGraphicsView* view = new QGraphicsView(&scene);
    CGAL::Qt::GraphicsViewNavigation navigation;
    view->installEventFilter(&navigation);
    view->viewport()->installEventFilter(&navigation);
    view->setRenderHint(QPainter::Antialiasing);
    view->show();
    return app.exec();
}
```

CmakeLists.txt in GraphicsView

- Open the second file `CMakeLists.txt` in the folder GraphicsView



```
# This is the CMake script for compiling a CGAL application.

project (GraphicsView_Demo)

cmake_minimum_required(VERSION 2.8.11)
if(POLICY CMP0053)
  cmake_policy(SET CMP0053 OLD)
endif()
if(POLICY CMP0043)
  cmake_policy(SET CMP0043 OLD)
endif()

find_package(CGAL COMPONENTS Qt5)

include(${CGAL_USE_FILE})

find_package(Qt5 QUIET COMPONENTS Xml Script OpenGL Svg)

if ( CGAL_FOUND AND CGAL_Qt5_FOUND AND Qt5_FOUND )

  add_definitions(-DQT_NO_KEYWORDS)

  add_executable  ( min min.cpp  ${CGAL_Qt5_RESOURCE_FILES} ${CGAL_Qt5_MOC_FILES})

  qt5_use_modules(min Xml Script OpenGL Svg)

  add_to_cached_list( CGAL_EXECUTABLE_TARGETS min )

  # Link with Qt libraries
  target_link_libraries( min ${QT_LIBRARIES} )
  # Link with CGAL
  target_link_libraries( min ${CGAL_LIBRARIES}  ${CGAL_3RD_PARTY_LIBRARIES})

else()

  message(STATUS "NOTICE: This demo requires CGAL and Qt5, and will not be compiled.")

endif()
```

Why CmakeLists.txt?

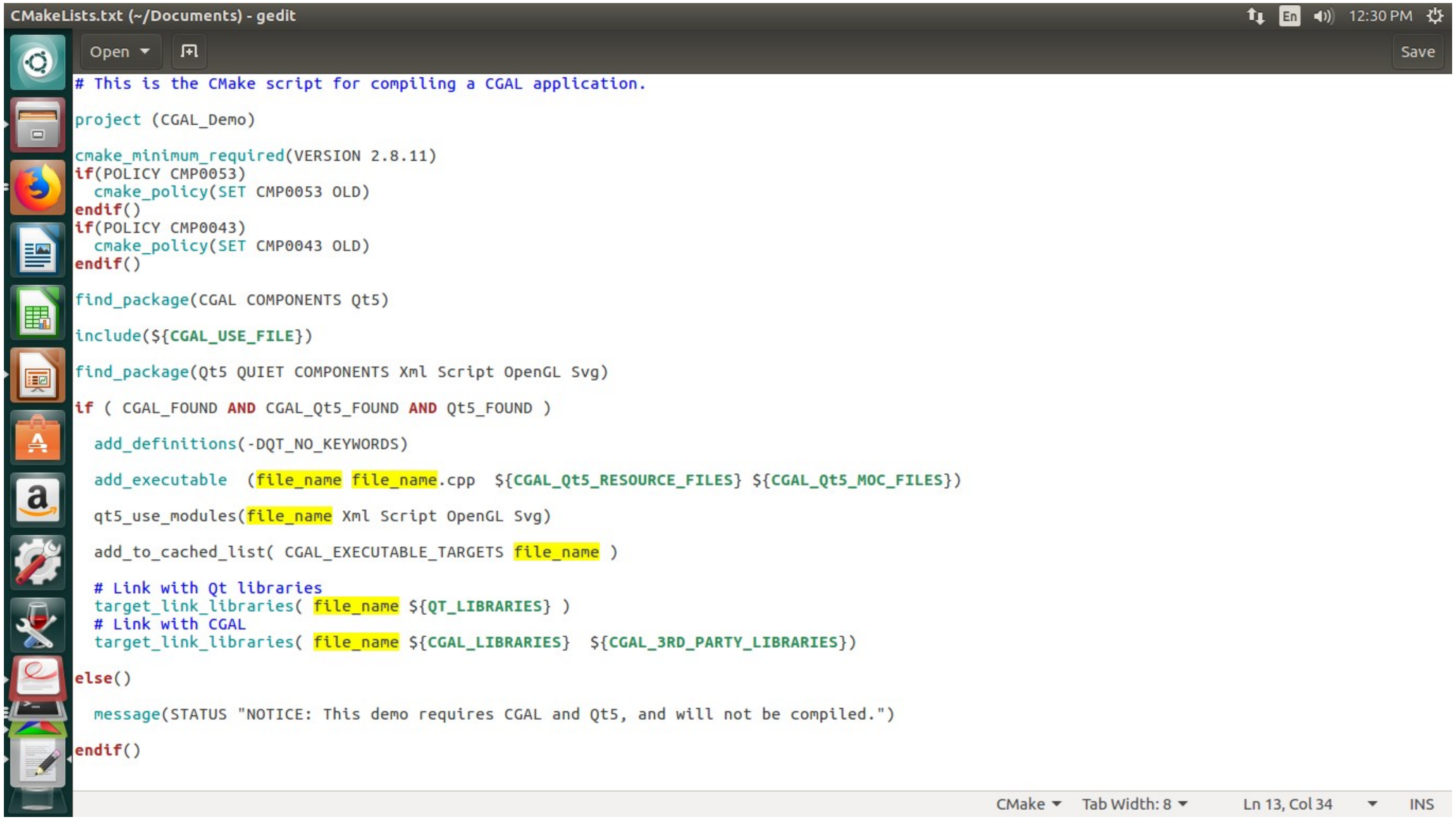
→ To link a custom program with CGAL

https://doc.cgal.org/5.2/Manual/devman_create_and_use_a_cmakelist.html

2. How to modify GraphicsView to create your own demo

1. In your Desktop, create a folder for your demo, say demo1
2. Create a folder named bin inside the folder demo1
3. Copy min.cpp from the folder GraphicsView to your folder demo1
4. Rename your source code to demo1.cpp (or any name)
5. Copy CMakeLists.txt from the folder GraphicsView to your folder demo1
6. Change the source code file name min (where ever it is present) in the CMakeLists.txt to demo1

Editing *'CMakeLists.txt'*



```
# This is the CMake script for compiling a CGAL application.

project (CGAL_Demo)

cmake_minimum_required(VERSION 2.8.11)
if(POLICY CMP0053)
  cmake_policy(SET CMP0053 OLD)
endif()
if(POLICY CMP0043)
  cmake_policy(SET CMP0043 OLD)
endif()

find_package(CGAL COMPONENTS Qt5)

include(${CGAL_USE_FILE})

find_package(Qt5 QUIET COMPONENTS Xml Script OpenGL Svg)

if ( CGAL_FOUND AND CGAL_Qt5_FOUND AND Qt5_FOUND )

  add_definitions(-DQT_NO_KEYWORDS)

  add_executable  (file_name file_name.cpp  ${CGAL_Qt5_RESOURCE_FILES} ${CGAL_Qt5_MOC_FILES})

  qt5_use_modules(file_name Xml Script OpenGL Svg)

  add_to_cached_list( CGAL_EXECUTABLE_TARGETS file_name )

  # Link with Qt libraries
  target_link_libraries( file_name ${QT_LIBRARIES} )
  # Link with CGAL
  target_link_libraries( file_name ${CGAL_LIBRARIES}  ${CGAL_3RD_PARTY_LIBRARIES})

else()

  message(STATUS "NOTICE: This demo requires CGAL and Qt5, and will not be compiled.")

endif()
```

Change the file_name(highlighted) to your .*cpp* file name and save it.

Compile & Execute your cpp file

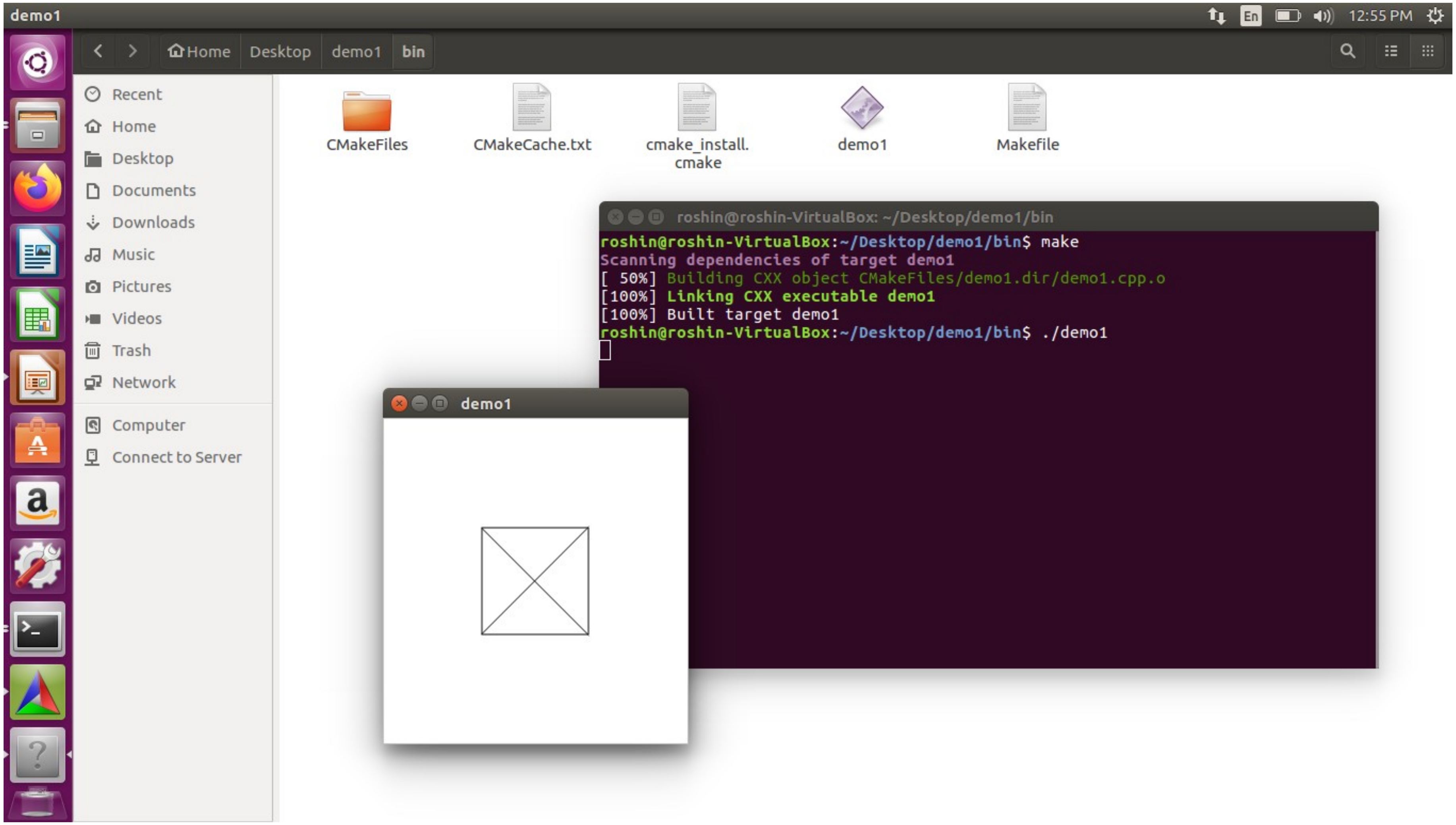
1. From bin folder, **cmake ~/Desktop/cgalDemo1**

2. Your makefiles are ready in bin folder now

3. From bin folder **make** to compile

4. Execute your program by clicking on the executable file or by typing *./demo1* in the terminal and you will get the Output displayed in QT window.

Execute your program by clicking on the executable file or by typing *./demo1* in the terminal and you will get the output displayed in QT window.

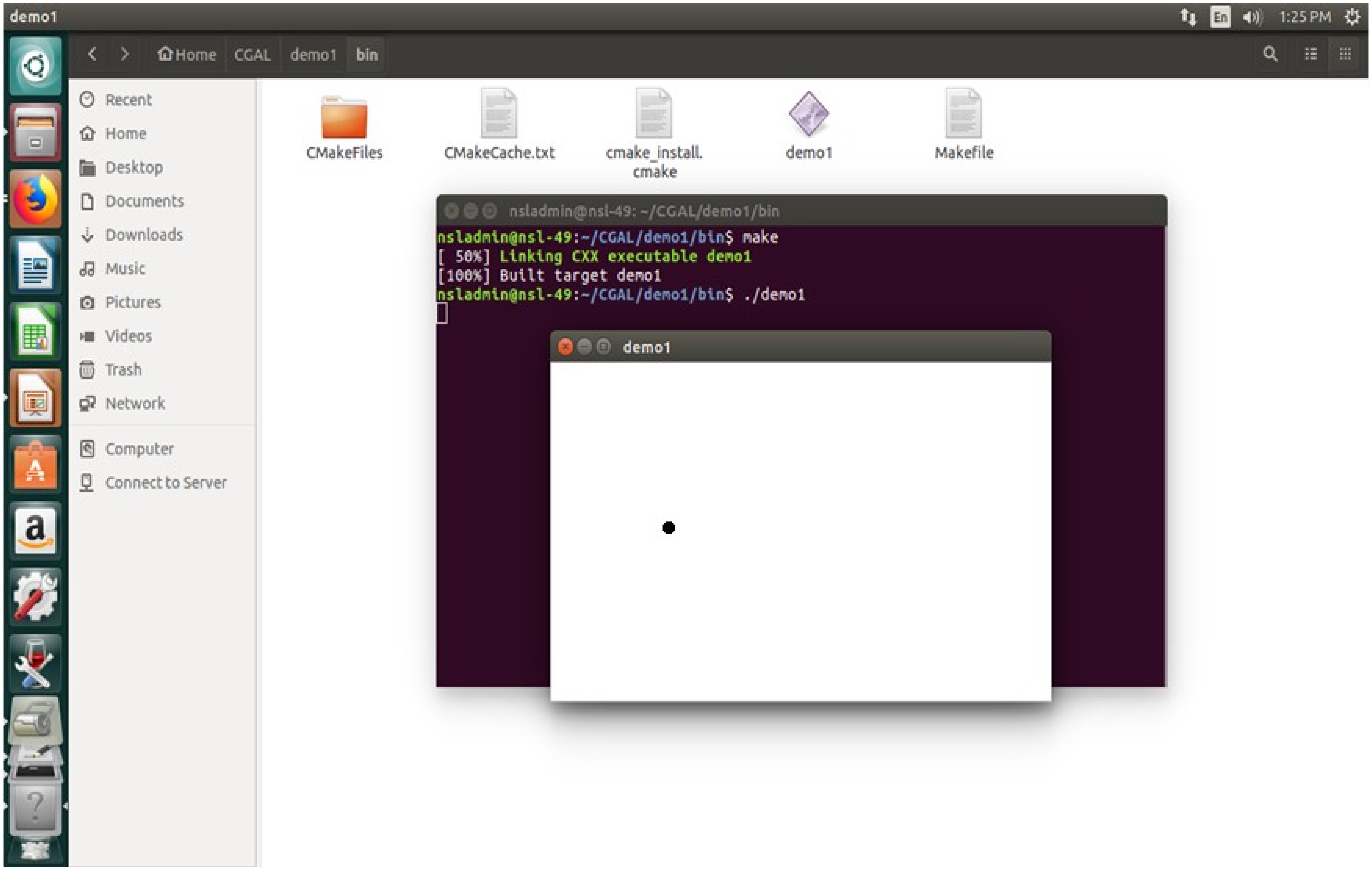


```
roshin@roshin-VirtualBox:~/Desktop/demo1/bin$ make
Scanning dependencies of target demo1
[ 50%] Building CXX object CMakeFiles/demo1.dir/demo1.cpp.o
[100%] Linking CXX executable demo1
[100%] Built target demo1
roshin@roshin-VirtualBox:~/Desktop/demo1/bin$ ./demo1
```

Display a point

```
#include <iostream>  // cpp header
#include <QtGui>     // QT header for GUI
#include <QApplication> // QT header for creating a QT GUI application
#include <QGraphicsScene> // QT header for 2D graphical items
#include <QGraphicsView> // QT header for displaying the contents of a scene
#include <CGAL/Qt/GraphicsViewNavigation.h>  //CGAL header for linking CGAL and
   QGraphicsView

int main(int argc, char **argv)
{
    QApplication app(argc, argv);  //create a new QT GUI application
    QGraphicsScene scene;         //create a QT scene to display point
    scene.setSceneRect(0,0, 500, 300);  //set scene size

   //add a point in the scene
    scene.addEllipse(100, 100, 5, 5);

   //display the scene
    QGraphicsView* view = new QGraphicsView(&scene);
    CGAL::Qt::GraphicsViewNavigation navigation;
    view->installEventFilter(&navigation);
    view->viewport()->installEventFilter(&navigation);
    view->setRenderHint(QPainter::Antialiasing);
    view->show();
    return app.exec();
}
```

Output



```
nsladmin@nsl-49:~/CGAL/demo1/bin$ make
[ 50%] Linking CXX executable demo1
[100%] Built target demo1
nsladmin@nsl-49:~/CGAL/demo1/bin$ ./demo1
```

Question: Read the x and y coordinates of a set of points from a text file *input.txt* and display the points on a QT window using CGAL.

1. In your Desktop, create a folder for your demo, say Points

2. Create a folder named bin inside the folder Points

3. Copy min.cpp from the folder GraphicsView to your folder Points

4. Rename your source code to points.cpp

5. Copy CMakeLists.txt from the folder GraphicsView to your folder Points

6. Change the source code file name min (where ever it is present) in the CMakeLists.txt to Points

7. Create input.txt inside the folder bin

8. How to change min.cpp to points.cpp ???

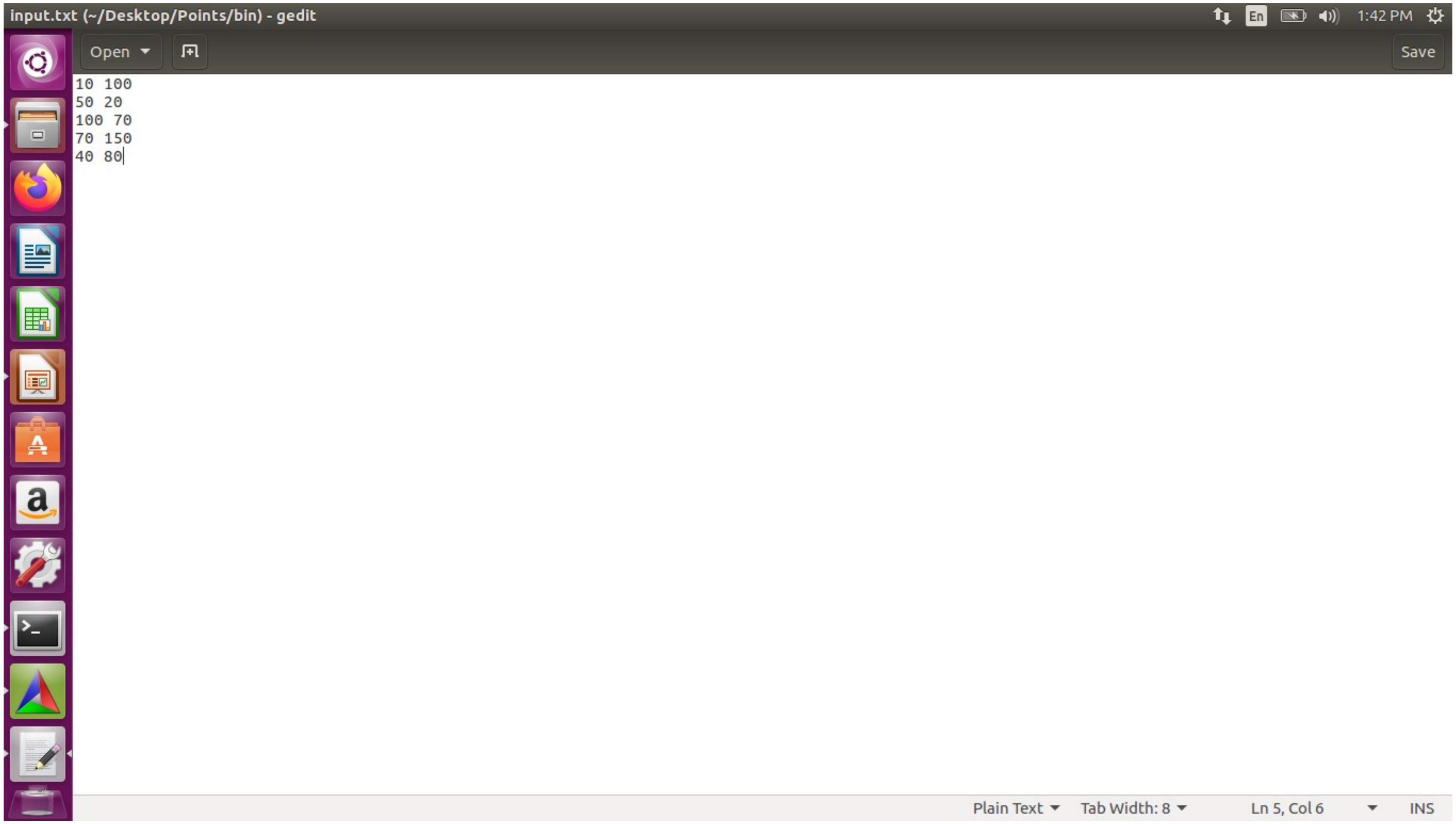
points.cpp

```
#include <iostream>
#include <fstream>  //cpp header for file handling
#include <QtGui>
#include <QApplication>
#include <QGraphicsScene>
#include <QGraphicsView>
#include <CGAL/Qt/GraphicsViewNavigation.h>
#include <QPen>  //QT header for colors and styles
#include <CGAL/Cartesian.h> // CGAL header for Cartesian coordinates
#include <CGAL/Point_2.h> // CGAL header for 2D points

//Define datatype Point_2
typedef CGAL::Cartesian<double> K;  // define type K as Cartesian coordinate
typedef K::Point_2 Point_2;     // define type Point_2 as 2D Cartesian point

int main(int argc, char **argv)
{
   Point_2 p; //Declare a variable of type Point_2
   std::ifstream iFile("input.txt", std::ios::in); //Open file for reading

   QApplication app(argc, argv);
   QGraphicsScene scene;
   scene.setSceneRect(0,0, 500, 300); //Set window size of your choice
```

Lines changed are in red color

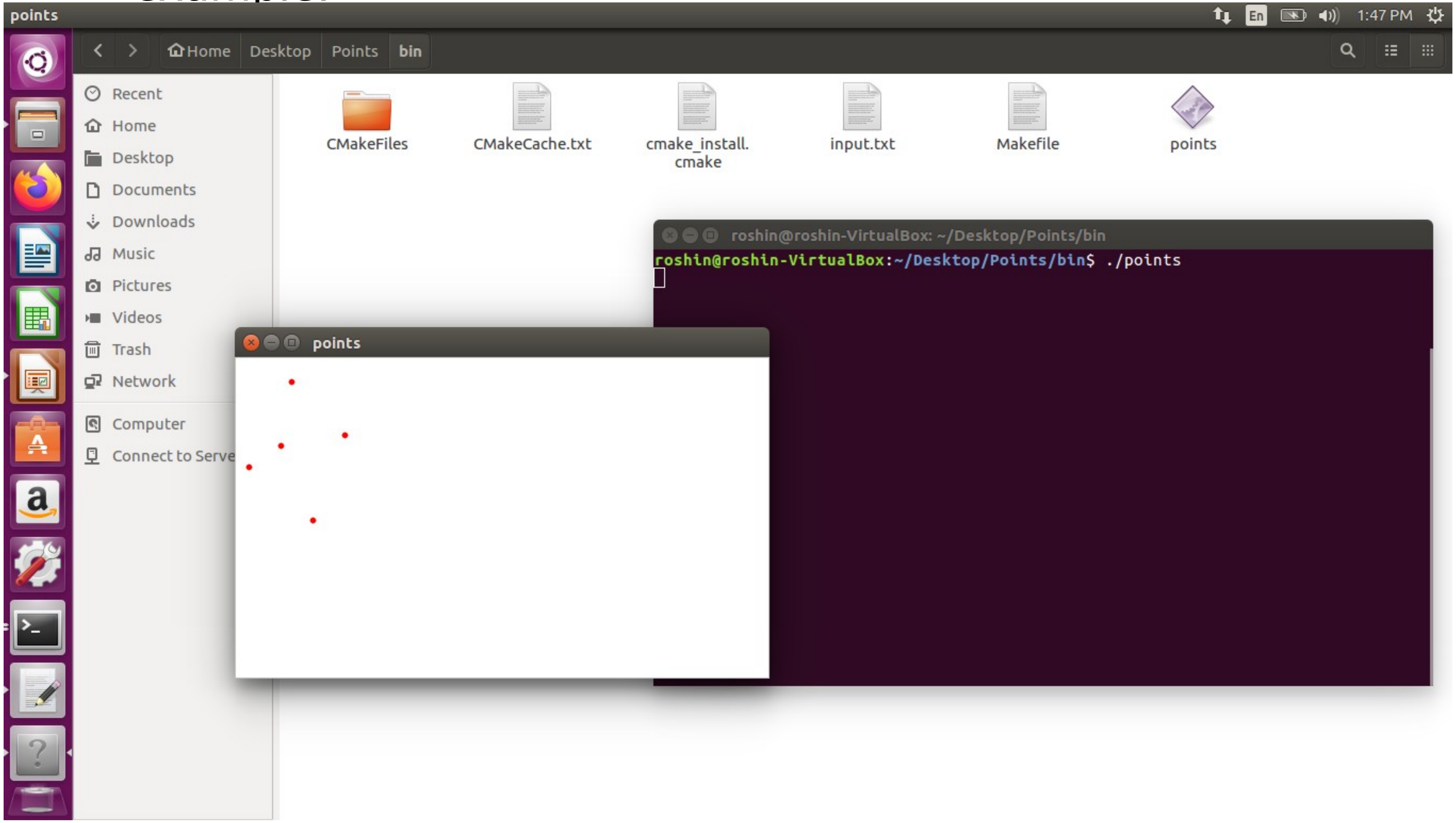
```
 QPen pen;  //Declare a variable of type Pen
 pen.setColor(Qt::red); // Set the color of pen as red

 while(iFile >> p)         //read input points into variable p
       scene.addEllipse(p.x(), p.y(), 5, 5,  pen, QBrush(Qt::red)); //add a
   point in the QT window by drawing a circle of small diamater


 QGraphicsView* view = new QGraphicsView(&scene);
 CGAL::Qt::GraphicsViewNavigation navigation;
 view->installEventFilter(&navigation);
 view->viewport()->installEventFilter(&navigation);
 view->setRenderHint(QPainter::Antialiasing);

 view->show();
 return app.exec();
}
```

input.txt



```
10 100
50 20
100 70
70 150
40 80
```

- make and run your demo as explained in the previous example.



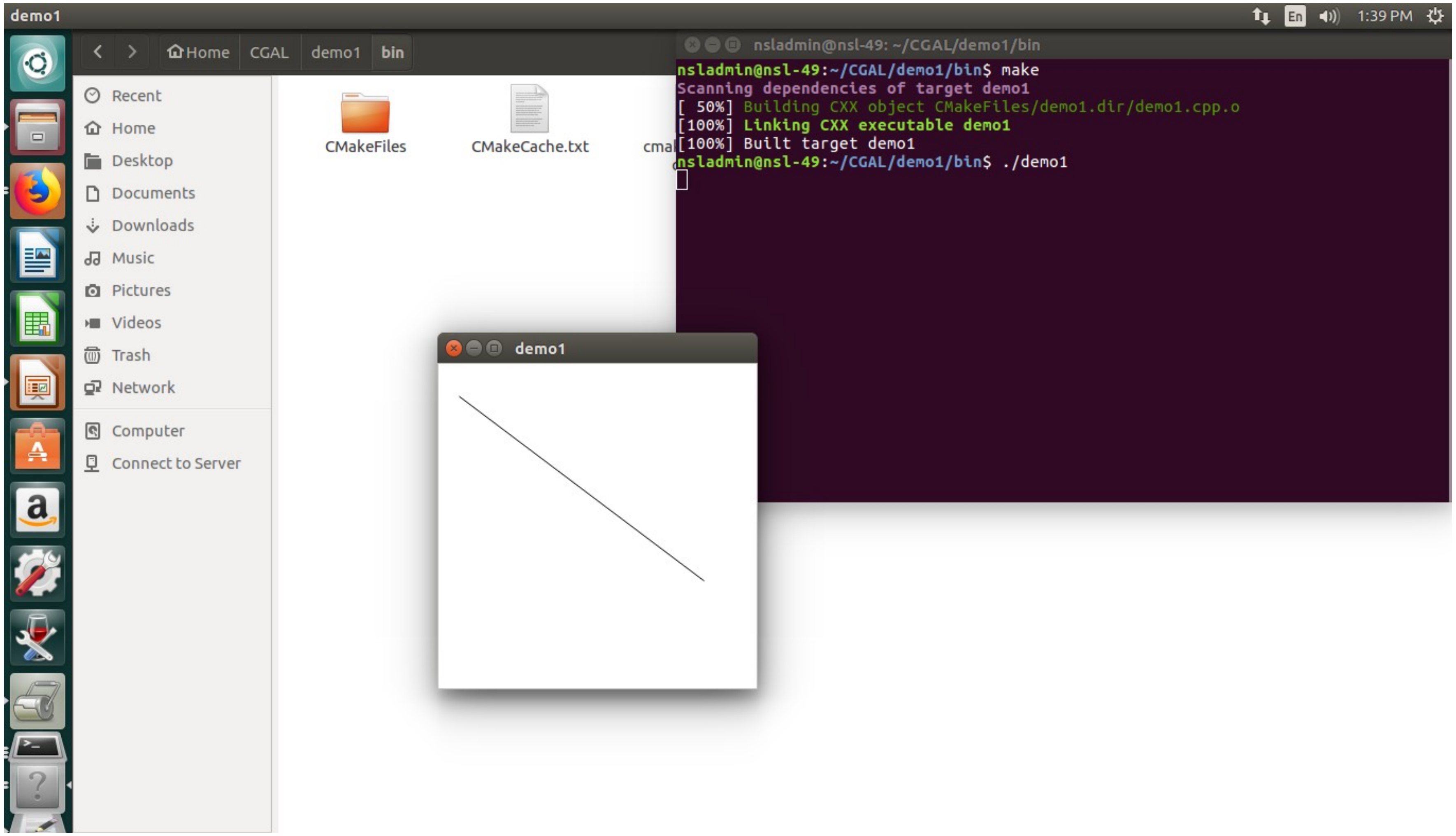
```
roshin@roshin-VirtualBox: ~/Desktop/Points/bin
roshin@roshin-VirtualBox:~/Desktop/Points/bin$ ./points
```

Some other examples

1. Display a line

```
#include <iostream>
#include <fstream>
#include <QtGui>
#include <QApplication>
#include <QGraphicsScene>
#include <QGraphicsView>
#include <CGAL/Qt/GraphicsViewNavigation.h>


int main(int argc, char **argv)
{
    QApplication app(argc, argv);

    QGraphicsScene scene;
    scene.setSceneRect(0,0, 300, 300);

    scene.addLine(QLineF(20, 30, 250, 200)); //add line to the scene

    QGraphicsView* view = new QGraphicsView(&scene);
    CGAL::Qt::GraphicsViewNavigation navigation;
    view->installEventFilter(&navigation);
    view->viewport()->installEventFilter(&navigation);
    view->setRenderHint(QPainter::Antialiasing);
    view->show();
    return app.exec();
}
```

Output



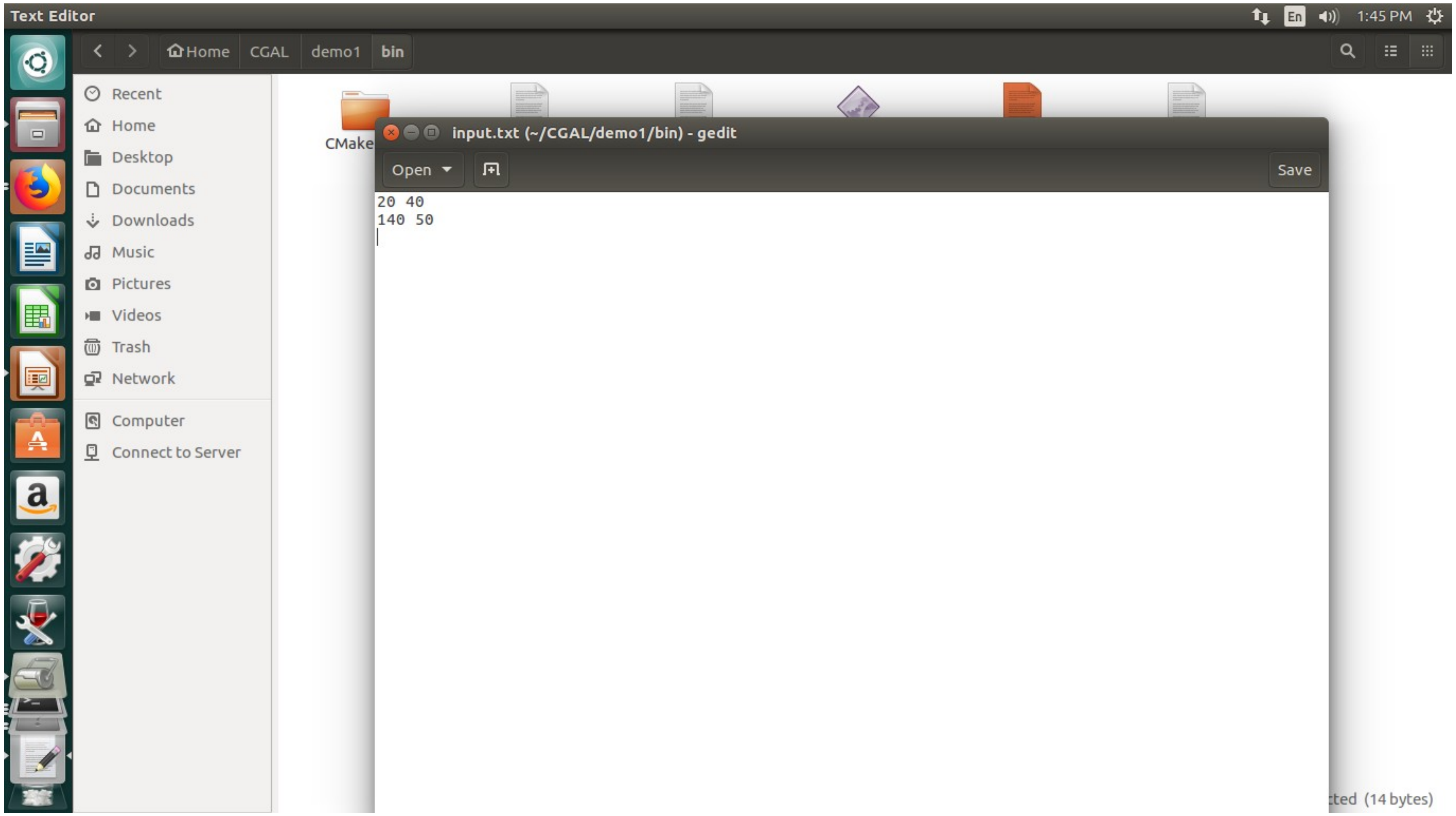
```
nsladmin@nsl-49:~/CGAL/demo1/bin$ make
Scanning dependencies of target demo1
[ 50%] Building CXX object CMakeFiles/demo1.dir/demo1.cpp.o
[100%] Linking CXX executable demo1
[100%] Built target demo1
nsladmin@nsl-49:~/CGAL/demo1/bin$ ./demo1
```

2. Display a line from input file

```cpp
#include <iostream>
#include <fstream>
#include <QtGui>
#include <QApplication>
#include <QGraphicsScene>
#include <QGraphicsView>
#include <CGAL/Qt/GraphicsViewNavigation.h>
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>

typedef CGAL::Cartesian<double> K;
typedef K::Point_2 Point_2;

int main(int argc, char **argv)
{

   std::ifstream iFile("input.txt", std::ios::in);

   Point_2 p1, p2;
```

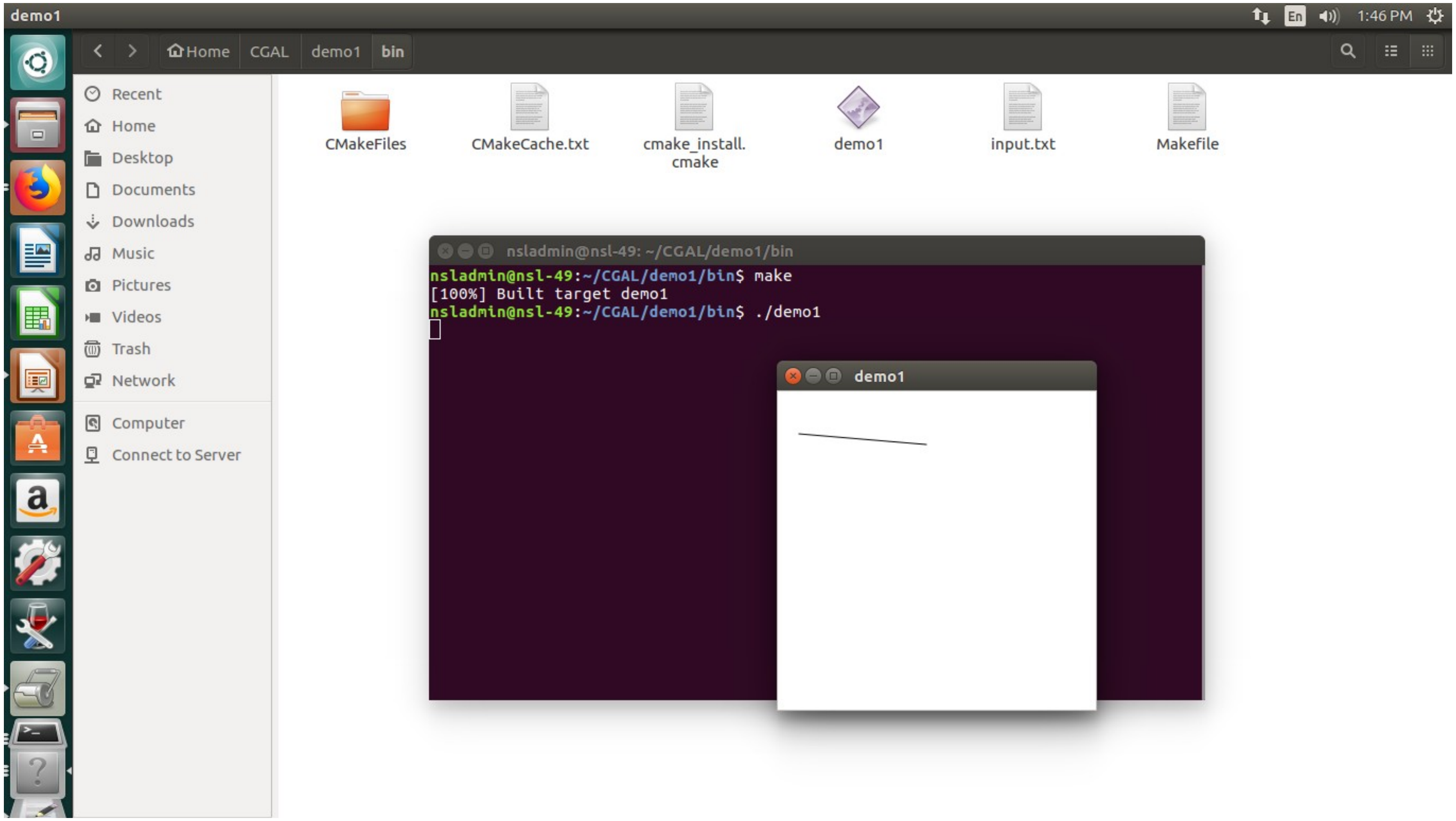
```
    iFile >> p1;
    iFile >> p2;

    QApplication app(argc, argv);
    QGraphicsScene scene;

    scene.setSceneRect(0,0, 300, 300);
    scene.addLine(QLineF(p1.x(),p1.y(),p2.x(),p2.y()));


    QGraphicsView* view = new QGraphicsView(&scene);
    CGAL::Qt::GraphicsViewNavigation navigation;
    view->installEventFilter(&navigation);
    view->viewport()->installEventFilter(&navigation);
    view->setRenderHint(QPainter::Antialiasing);
    view->show();
    return app.exec();
}
```

Input



```
20 40
140 50
```

Output

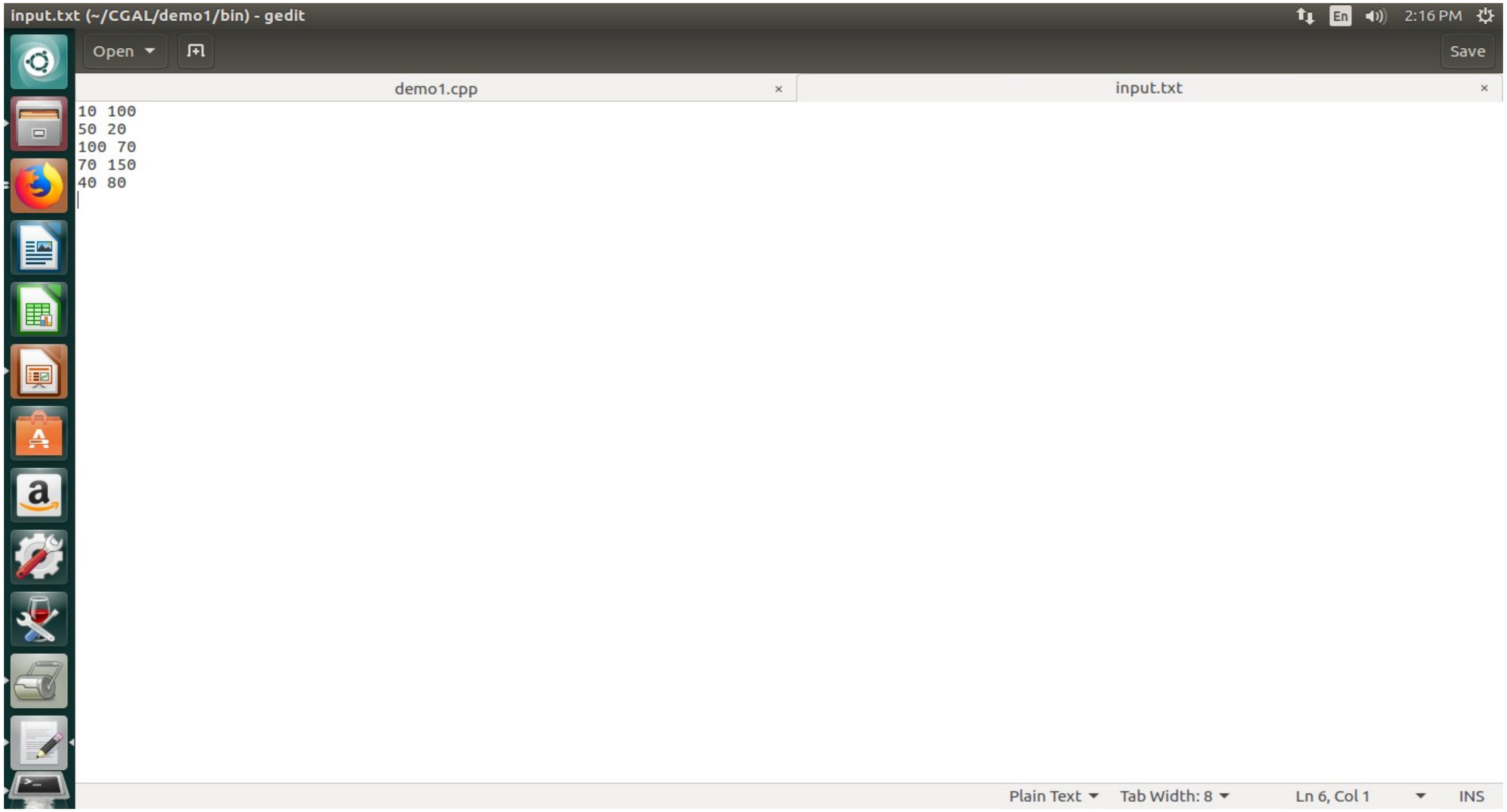


```
nsladmin@nsl-49:~/CGAL/demo1/bin$ make
[100%] Built target demo1
nsladmin@nsl-49:~/CGAL/demo1/bin$ ./demo1
```

3. Display a polygon in QT window

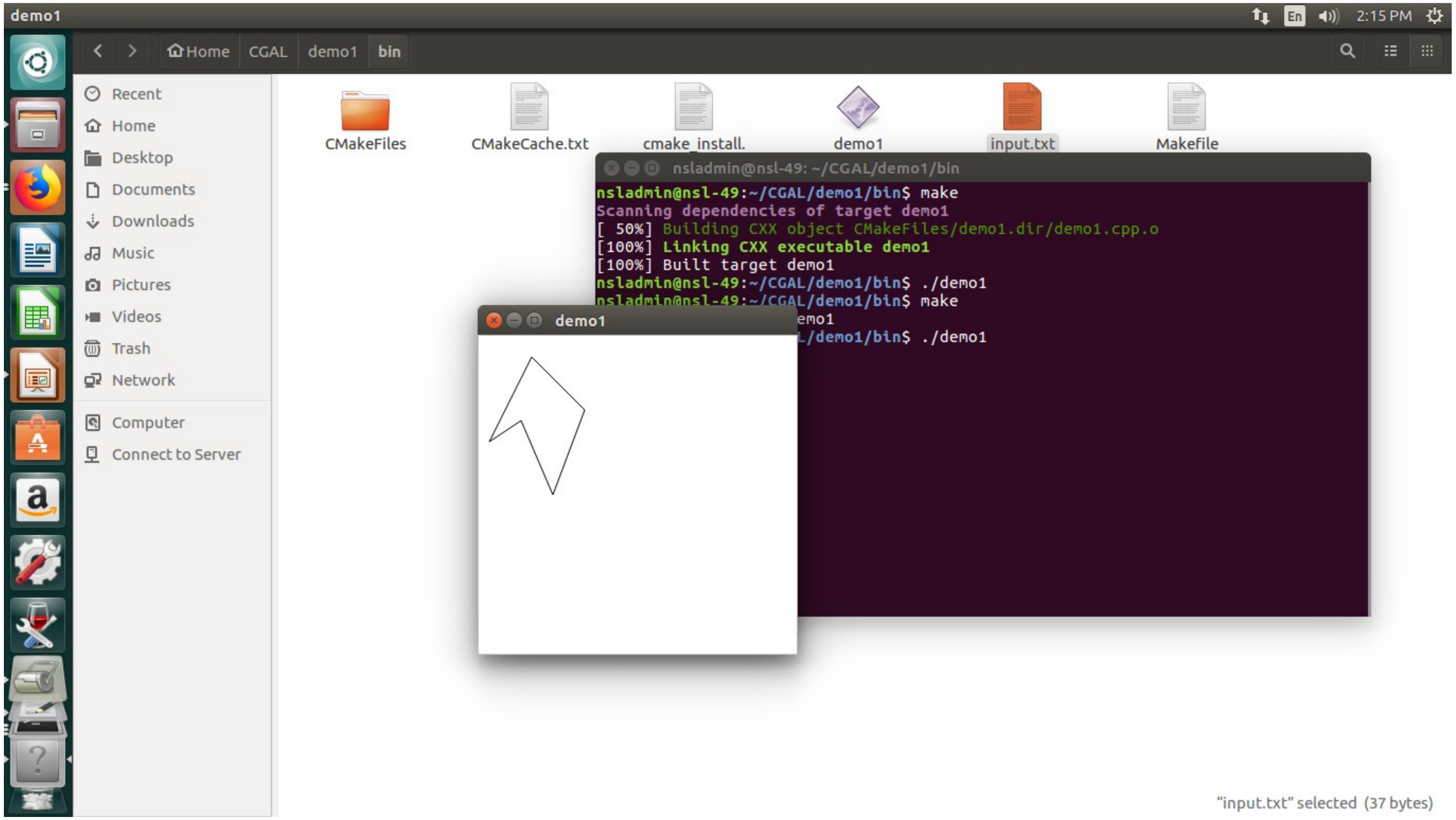
```
#include <iostream>
#include <fstream>
#include <QtGui>
#include <QApplication>
#include <QGraphicsScene>
#include <QGraphicsView>
#include <QPolygonF>
#include <CGAL/Qt/GraphicsViewNavigation.h>
#include <CGAL/Cartesian.h>
#include <CGAL/Point_2.h>

typedef CGAL::Cartesian<double> K;
typedef K::Point_2 Point_2;

int main(int argc, char **argv)
{
    std::ifstream iFile("input.txt", std::ios::in);
    Point_2 p1;

    QApplication app(argc, argv);

    QGraphicsScene scene;
    scene.setSceneRect(0,0, 300, 300);
    QPolygonF polygon;

 //store coordinates in QT polygon
    while (iFile >> p1)
        polygon << QPointF(p1.x(), p1.y());

 //add polygon to the QT scene
    scene.addPolygon(polygon);

    QGraphicsView* view = new QGraphicsView(&scene);
    CGAL::Qt::GraphicsViewNavigation navigation;
    view->installEventFilter(&navigation);
    view->viewport()->installEventFilter(&navigation);
    view->setRenderHint(QPainter::Antialiasing);
    view->show();
    return app.exec();
}
```

Input



```
10 100
50 20
100 70
70 150
40 80
```

Output



```
nsladmin@nsl-49:~/CGAL/demo1/bin$ make
Scanning dependencies of target demo1
[ 50%] Building CXX object CMakeFiles/demo1.dir/demo1.cpp.o
[100%] Linking CXX executable demo1
[100%] Built target demo1
nsladmin@nsl-49:~/CGAL/demo1/bin$ ./demo1
nsladmin@nsl-49:~/CGAL/demo1/bin$ make
emo1
L/demo1/bin$ ./demo1
```

Practice Questions

1. Read a text file of 2D points and draw multiple line segments.

2. Write a program to find given (multiple) line segments intersect or not and find the points of intersection. Display whether each pair of line segments intersects (Yes/ No) and find the point of intersection for each pair of line segments.

3. Write a program to find out the area of a polygon, color each triangles of the polygon using different colors.

Enjoy Using CGAL