Guía QGraphicsView, QGraphicsScene, QGraphicsItem.

Qt cuenta con tres clases esenciales para la correcta implementación de la simulación de los sistemas físicos que se desarrollan en el curso.

1. **Qué es vista, escena y elementos? Con qué clases se implementan?**

Para realizar simulaciones físicas se necesitan tres elementos básicos en Qt, los elementos son los objetos que están sometidos a las leyes físicas, la escena es lugar donde se mueven estos elementos y la vista es donde podemos observar la interacción de los elementos en un entorno (escena).

Los elementos son figuras geométricas o personalizadas de dos dimensiones. Los elementos deben responder ante movimientos de mouse y teclado, es posible arrastrar y soltar, detectar colisiones y agrupar. Se implementan los elementos por medio de la clase QGraphicsItem, es la clase base para los elementos gráficos de un QGraphicsScene de la escena. QGraphicsItem cuenta con métodos que permiten definir la geometría, detectar colisiones, cómo se ven e interactuar con otros elementos. Toda la información geométrica del elemento se basa en un sistema de coordenadas locales.

Un ejemplo de los elementos se observa en la figura 1. Figuras geométricas, esferas con radio variable.

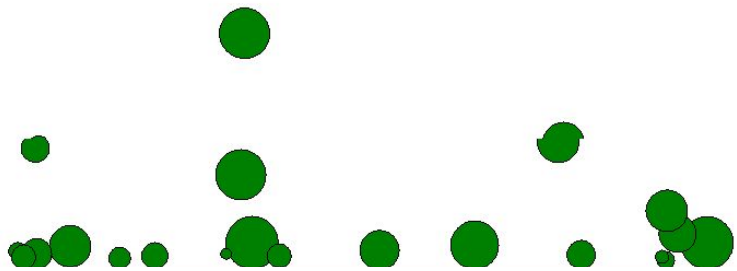

Figura 1. Los elementos

La escena es el espacio que contiene los elementos, es mostrada por la vista (view). La escena debe proveer una interfaz rápida para manejar un gran número de elementos, propagar los eventos a cada elemento, manejar cada elemento y permitir mantener la funcionalidad entre la escena y los elementos. La clase QGraphicsScene permite implementar la escena. Proporciona una superficie para manejar un gran número de elementos 2D. La clase sirve como contenedor de elementos y junto a la vista permite visualizarlos.

La figura 2 tenemos la escena, al implementarla permite contener los elementos y la interacción entre ellos y la escena.

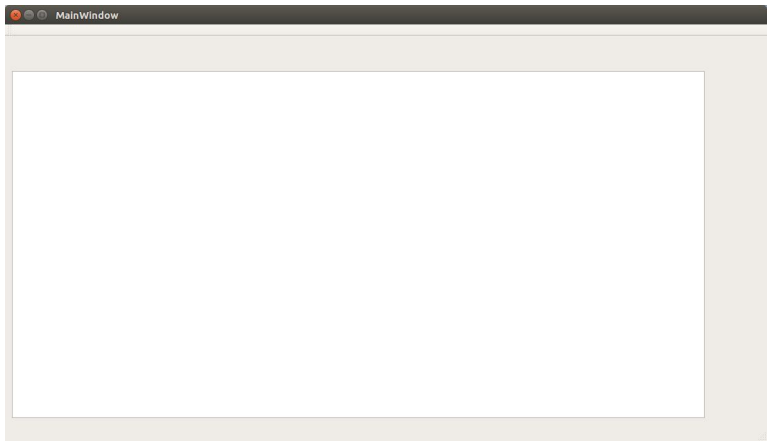

Figura 2. La escena

La interfaz gráfica de usuario GUI, en Qt permite agregar gran cantidad de elementos, uno de ellos es Graphics View donde se muestra el contenido de la escena. Graphics View es un objeto de la clase QGraphicsview. En la figura 3 se agrega a la vista la escena y a su vez los elemento a la escena. La vista nos permite observar los elementos y la interacción en la escena.

Figura 3. La vista.

2. Sistema de Coordenadas en Qt.

La ubicación de los elementos en la escena y visualización en el view depende del sistema de coordenadas. En la figura 1 vemos el sistema de coordenadas de dos dimensiones en Qt. Nótese que el eje y está invertido.

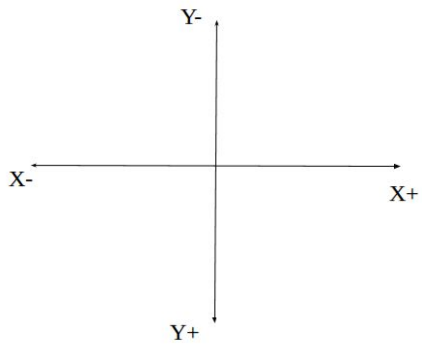

Figura 4: Sistema de coordenadas en Qt.

2.1 Sistema de Coordenadas del graphics View:

Graphics view se basa en un sistema de coordenadas cartesianas, la posición de los elementos y la geometría de la escena depende de las coordenadas x,y. Una unidad en la escena está representada por un pixel en la pantalla.

Existe un sistema de coordenadas para los elementos, la escena y la vista , a continuación se describen sus principales características.

2.1.1 Sistema de coordenadas de elemento:

Los ítem tienen su propio sistema de coordenadas local. El origen (0,0) suele estar ubicado en el centro del ítem.

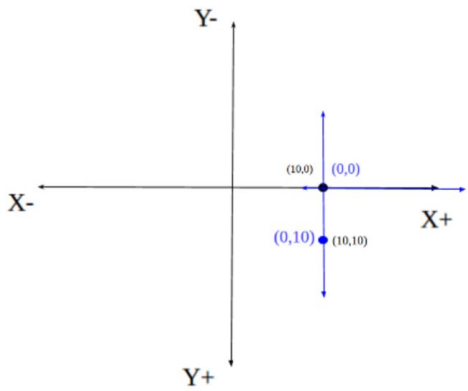

Figura 5: Sistema de coordenadas de la **escena** y el **ítem**.

El sistema de referencia del ítem es relativo a la escena, si el origen del elemento está ubicado en la misma posición del origen de la escena, los sistemas son idénticos. En cambio, si el origen del elemento se ubica en una posición diferente, como se ve en la figura 2, en el punto **(10,0)** en el sistema de la escena. El punto **(0,10)** en el sistema de coordenadas del ítem, será para la escena el punto **(10,10).**

QGraphicsItem provee métodos que la mayoría trabajan en coordenadas locales, como es lo boundingRect(), a excepción de pos(), que devuelve coordenadas de la escena.

2.1.2. Coordenadas de la escena.

La escena representa el sistema de coordenadas base para los elementos. Describe la posición de los ítem a nivel superior. Cada elemento está delimitado por el boundigRect y posición dentro de la escena.

2.1.3 Coordenadas del view:

Cada pixel representa un punto en el sistema de coordenadas del view. La esquina superior izquierda del widget de graphics view es es (0, 0), y la esquina inferior derecha siempre es (ancho del view , altura del view).

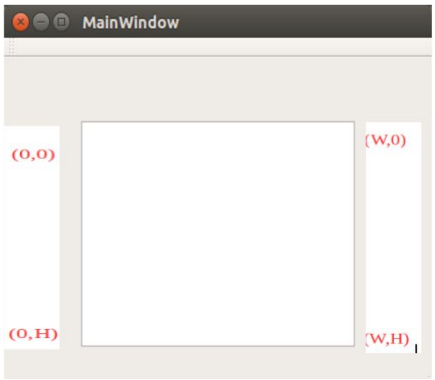

Figura 6: Sistema de coordenadas de la vista.

Actividad:

Implementar un sistema de cuatro esferas de diferente tamaño, ubicadas cerca de las esquinas de la escena. La escena será definida con dos sistemas de coordenadas diferentes, con el objetivo de variar el origen de coordenadas.

1. Vamos a crear un nuevo proyecto con interfaz gráfica, hacemos click en File->New File or Project->Qt Widget Project->Qt Gui Application. Seguir las instrucciones

2. En el fichero .ui agregamos widget Gaphics View. Display Widgets -> Graphics View.

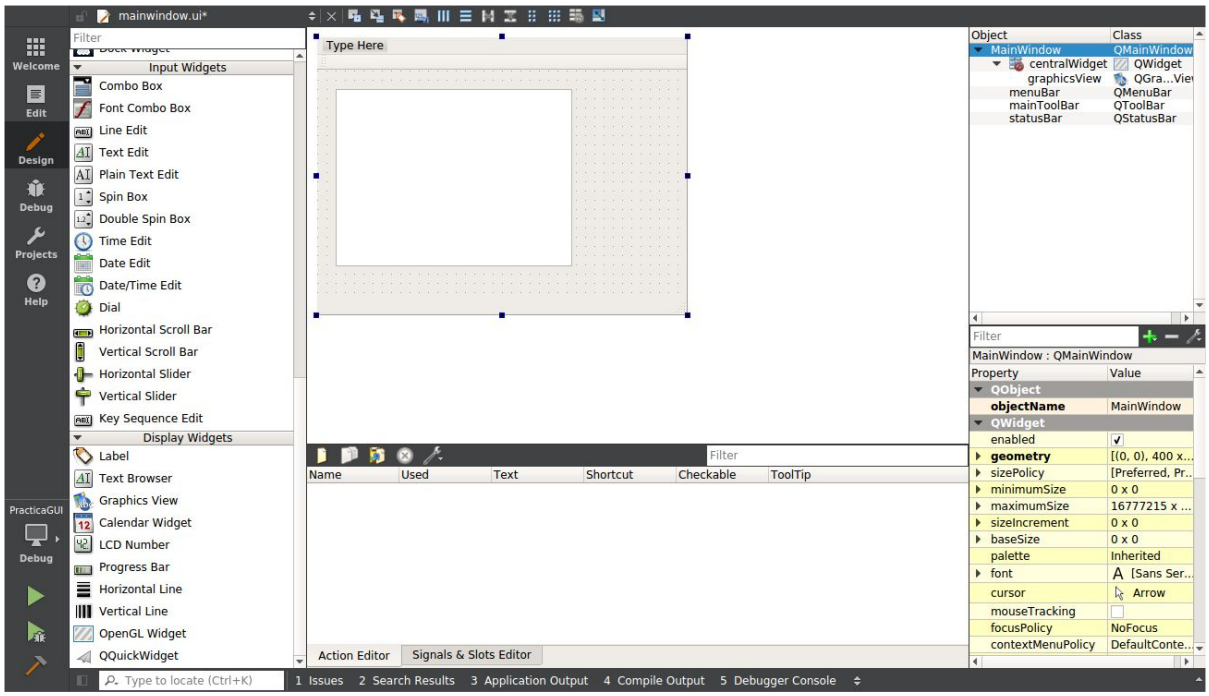

Figura 7: Agregando un Graphics View

3. Vamos agregar elementos a la escena, esferas. Necesitamos crear una nueva clase para poder gŕaficaras, debemos redefinir las funciones boundingRect y paint, además variables radio, posición en x y en y. La nueva clase debe heredar de QGraphicsItem.

   Redefinimos la función boundingRect, la función se encarga de delimitar un rectángulo, todo el elemento estará dentro de este cuadrado. La clase view utiliza ésta función para reservar el espacio necesario que ocupa el elemento en la escena. La forma del objeto puede ser arbitraria pero delimitada por el boundingRect.

```
QRectF boundingRect() const;
```

   Como la función retorna un rectángulo, vamos a utilizar la función QRectF que construye un rectángulo. Los parámetros que recibe son las coordenadas (x,y) para ubicar la esquina superior izquierda del rectángulo en la escena, el ancho y la altura del rectángulo.

```
QRectF Esfera::boundingRect() const
{
    return QRectF(x,y,2*radio,2*radio);
}
```

   Luego definimos la forma de nuestros elementos utilizando la función paint. Esta función utiliza coordenadas locales, debemos usar coordenadas que estén dentro de

las dimensiones del boundingRect. Podemos dibujar varios elementos, Qt cuenta con varias funciones para figuras geométricas. En nuestro caso vamos a dibujar una esfera, usamos la función drawEllipse de la clase QPainter, recibe como argumento las dimensiones del boundingRect.

Declaración y definición de la clase.

```
void paint(QPainter *painter,
           const QStyleOptionGraphicsItem *option, QWidget *widget);


void Esfera::paint(QPainter *painter,
                   const QStyleOptionGraphicsItem *option, QWidget *widget){
    painter->setBrush(Qt::darkGreen);
    painter->drawEllipse(boundingRect());
}
```

Es posible también indicar posición y tamaño de la esfera. Los dos primeros parámetros son las coordenadas locales (x,y) donde comienza el rectángulo, el ancho y el alto de la esfera, respectivamente.

```
painter->drawEllipse(-10, -10, 20, 20);
```

La función setBrush determina el color del elemento, Qt cuenta con variedad de colores.

Como vamos a ubicar varias esferas con diferente forma y posición, se debe definir el constructor de la clase de manera que reciba el radio, y posición de la esfera en la escena.

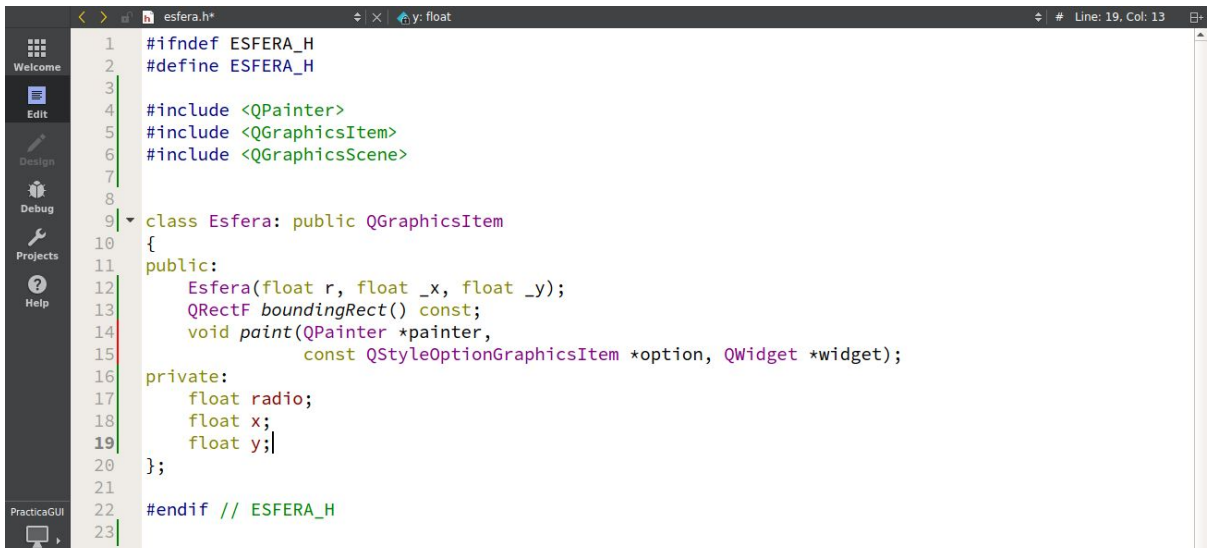

Figura 8: Definición de la clase Esfera

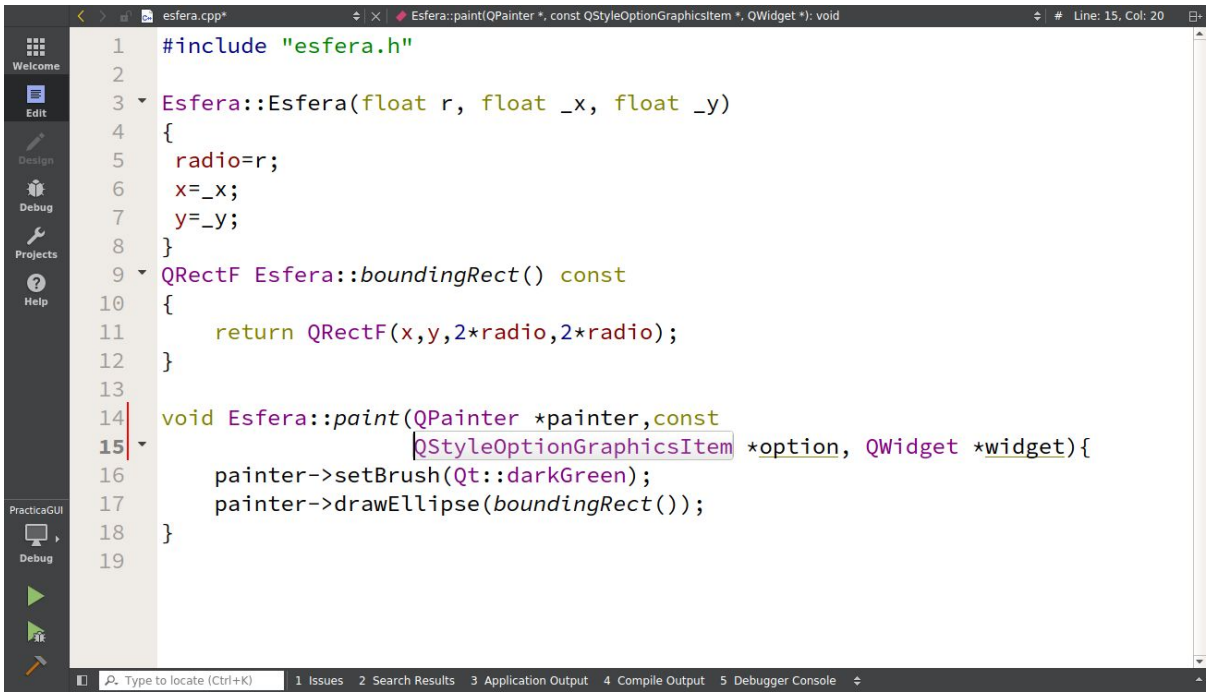

Figura 9: Definición de la clase Esfera.

4. En el fichero mainwindow.h definimos un puntero tipo QGraphicsScene para implementar la escena. Luego en el fichero mainwindow.cpp definimos su tamaño 400 * 400 (ancho, alto)

```
scene->setSceneRect(-200,-200,400,400);
```

Los dos primeros parámetros son las coordenadas (x,y) de la esquina superior izquierda del rectángulo que forma la escena, los dos últimos son el ancho y el alto respectivamente. El origen de coordenadas se encuentra en el centro de la escena.

Luego ingresamos la escena al view y a la ventana.

```
ui->graphicsView->setScene(scene);
```

Finalmente, nos queda agregar los elementos, usamos la función addItem, recibe como argumento un puntero del tipo elemento, por lo que necesitamos declarar un puntero tipo Esfera (La clase que se implementó en el numeral 3.)

Vamos a ubicar cuatro esferas de diferente tamaño cerca a las esquinas de la escena.

```
esfera[0]= new Esfera(10,-110,-110); //Esfera superior izquierda
scene->addItem(esfera[0]);
esfera[1]= new Esfera(20,80,-120); //Esfera superior Derecha
scene->addItem(esfera[1]);
esfera[2]= new Esfera(30,70,70); //Esfera inferior derecha
scene->addItem(esfera[2]);
esfera[3]= new Esfera(40,-140,60); //Esfera inferior izquierda
scene->addItem(esfera[3]);
esfera[4]= new Esfera(5,-5,-5); // Esfera del centro
scene->addItem(esfera[4]);
```

Los argumentos que recibe el constructor de la clase espera es el radio y posición (x,y) en la escena.

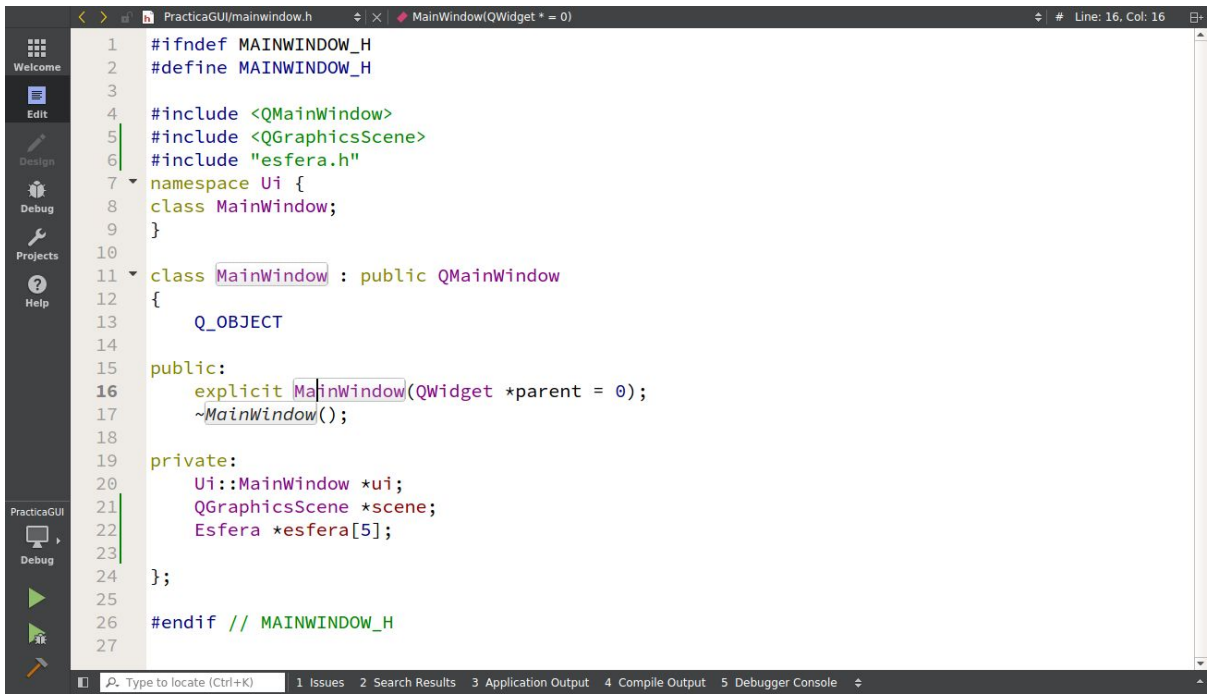

Figura 10: Declaración de la clase MainWindow

```cpp
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    scene=new QGraphicsScene(this);

    scene->setSceneRect(-200,-200,400,400);
    ui->graphicsView->setScene(scene);

    esfera[0]= new Esfera(10,-110,-110); //Esfera superior izquierda
    scene->addItem(esfera[0]);
    esfera[1]= new Esfera(20,80,-120); //Esfera superior Derecha
    scene->addItem(esfera[1]);
    esfera[2]= new Esfera(30,70,70); //Esfera inferior derecha
    scene->addItem(esfera[2]);
    esfera[3]= new Esfera(40,-140,60); //Esfera inferior izquierda
    scene->addItem(esfera[3]);
    esfera[4]= new Esfera(5,-5,-5); // Esfera del centro
    scene->addItem(esfera[4]);
```

Figura 11: Definición de la clase MainWindow

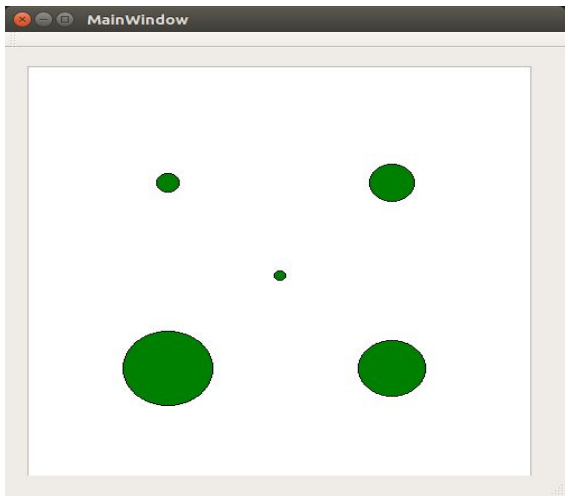

Figura 12: Salida del programa.

5. Ahora cambiemos las coordenadas de la escena en el view, vamos a ubicar esferas en la nueva escena.

   La esquina superior izquierda ya no será (-200,-200) , la cambiamos para que el origen esté en la esquina.

```
scene->setSceneRect(0,0,400,400);
```

Ya en la escena no podemos ubicar coordenadas negativas, si lo hiciéramos los elementos quedarían ubicados fuera de la escena y no sería posible visualizarlos.

Cambiamos las coordenadas para obtener las posiciones idénticas al numeral 4, teniendo en cuenta las nuevas coordenadas de la escena. Observe que en el centro de la escena ya no se encuentran en las coordenadas origen (0,0), si no en las coordenadas (200,200)

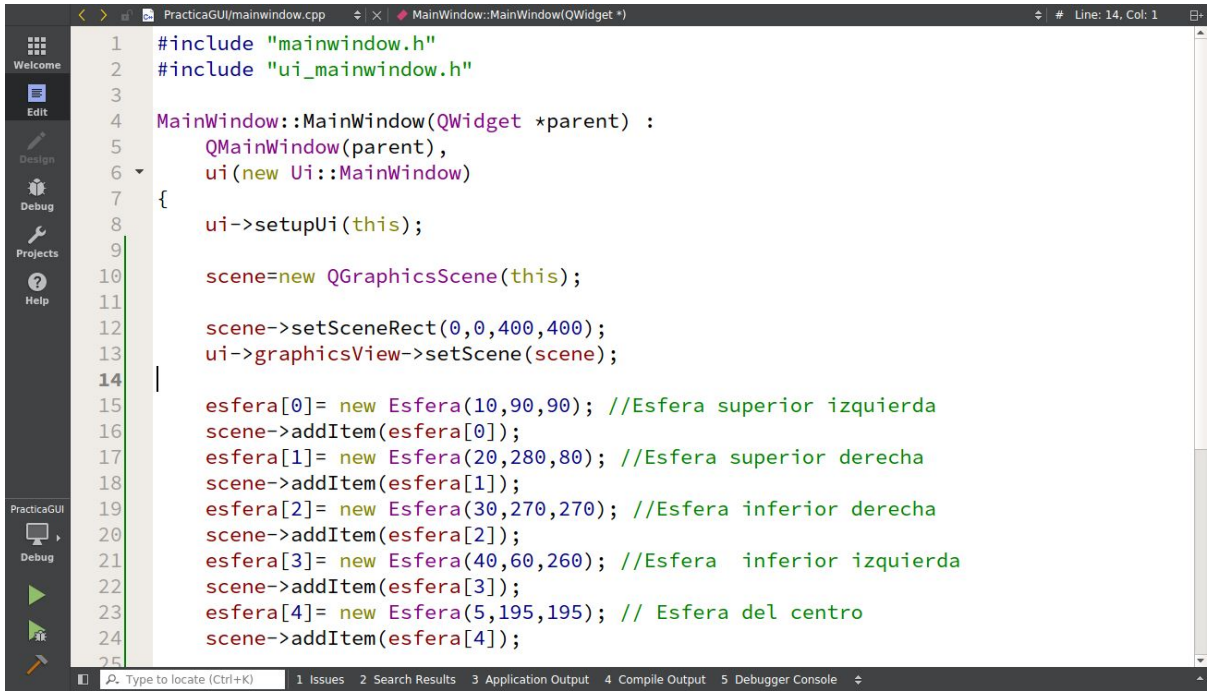

Figura 13: Nueva definición de la clase MainWindow

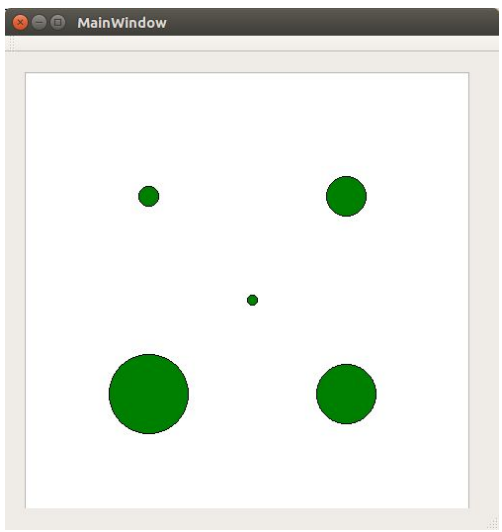

Figura 14: Salida del programa modificado

6. Realice siguiendo el procedimiento anterior un nuevo programa. Elija una figura geométrica que será su elemento usted diferente a la esfera, cambie el color, tamaño y ubicarlos siguiendo una forma de cruz en ambas variaciones del sistema de coordenadas explicadas en la guía. La escena debe tener medidas 600x400.

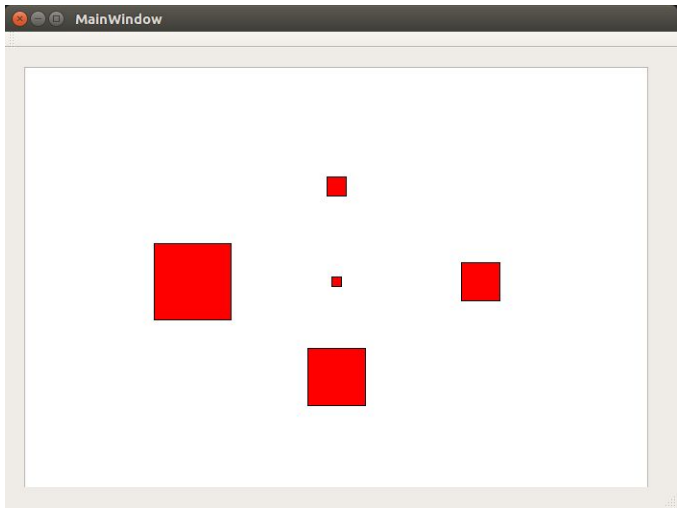

Figura 15: Ejemplo.

Documentación:

- **Sistema de coordenadas de The Graphics view:** http://doc.qt.io/qt-4.8/graphicsview.html#the-graphics-view-coordinate-system
- **Clase QPainter:** http://doc.qt.io/qt-4.8/qpainter.html
- **Clase QGraphicsItem:** http://doc.qt.io/qt-4.8/qgraphicsitem.html
- **Clase QGraphicsScene:** http://doc.qt.io/qt-4.8/qgraphicsscene.html
- **Clase QGraphicsView:** http://doc.qt.io/qt-4.8/qgraphicsview.html

Esta guía fue desarrollada por Sara Pavas con la supervisión de Augusto Salazar