ISSN:2790-1688 DOI: 10.56028/aetr.4.1.189.2023

Research and Application of Qt Graphic View Framework

Junfeng Zhu [1, a]

[1]Hangzhou Vango Technologies,Inc, Hangzhou, China.

[a] zhujf_lg5a_104@163.com

Abstract. Qt is a cross platform C++GUI application development framework. It can develop both GUI programs and non GUI programs. Qt is an object-oriented framework that is easy to extend and allows true component programming. Qt provides the Graphics View Framework to realize more advanced graphic applications. Using this framework, we can quickly design dynamic GUI applications and various drawing, collision detection and other applications.

Keywords：Qt；Image; Vector diagram; GUI; Industrial monitoring; robot；Laser mapping.

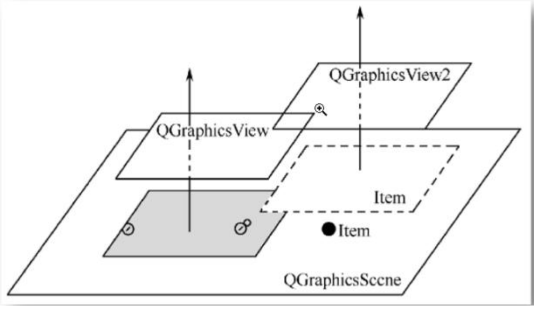
1. Introduction

For a simple drawing mechanism, you can use QPainter in the Qt library to implement ordinary two-dimensional graphics drawing. This method is to draw in the paintEvent event, which draws a bitmap image, which is suitable for uncomplicated drawing environments. The disadvantage is that you cannot select, edit, drag, etc. on the image.

The Graphics View Framework provides a way to manage a large number of custom 2D graphics items, interact through view widgets, and support scaling and rotation. Under this framework, including an event propagation architecture, the items in the scene have precise interaction capabilities (double precision level), and have keystrokes, mouse handling mechanisms, and the ability to track mouse movements. The Graphics View framework uses BSP (Binary Space Partitioning, or Binary Space Partitioning), which enables fast searches, even in scenes with millions of items.

2. Graphic View Architecture

The Graphics View framework mainly includes a scene class (QGraphicsScene), a view class (QGraphicsView) and a primitive class (QGraphicsItem). The scene class provides a container for managing many primitives located in it, the view class is used to display the primitives in the scene, a scene can be represented by multiple views, and a scene includes multiple geometric figures. The relationship between the three can be represented by the following figure:



The three-element relationship of the graphic view frame

2.1 The Scene

QGraphicsScene provides the Graphics View scene. The scene has the following responsibilities:

Providing a fast interface for managing a large number of items

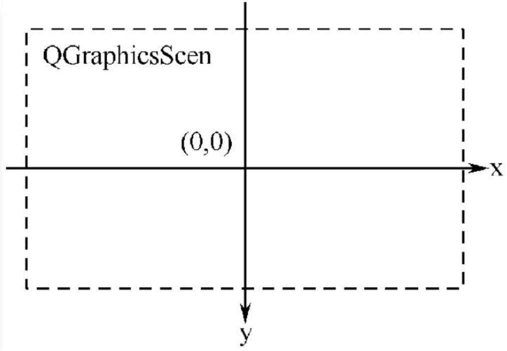
Propagating events to each item

Managing item state, such as selection and focus handling

Providing untransformed rendering functionality; mainly for printing

The scene is used as a container of primitives. By calling the add interface, different primitives are added to the scene, and then the functions in the scene are called to retrieve them. The event delivery architecture in the scene will dispatch events for primitives and manage event propagation between primitives. For example, the scene receives a mouse down event at a certain position, and the scene can pass the event to any primitive at that position. The scene also supports render functions to render parts of the scene to the drawing device.

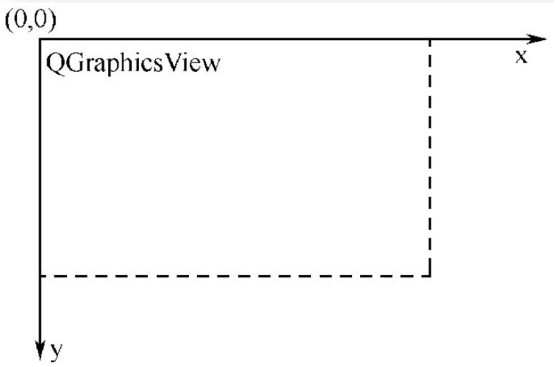
The scene coordinate system takes the center as the origin (0,0), as shown below



Scene coordinate system

2.2 The View

QGraphicsView provides view widgets whose role is to visualize the content of the scene. We can attach multiple views to the same scene to provide different presentation views of the same dataset. A view widget is a scrollable area that uses scroll bars for navigation in large scenes. The view widget supports OpenGL, and the QOpenGLWidget can be used as the viewport by calling the QGraphicsView::setViewport() function. Mouse and keyboard events are sent to the view, the view coordinates are converted into scene coordinates, and passed to the scene, and finally used in the visualization scene. Using the view's transformation matrix, the view coordinates can be converted to scene coordinates, allowing the implementation of advanced functions such as scaling and rotation.



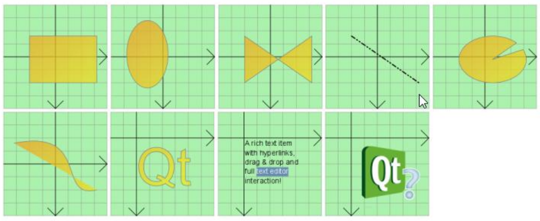
View Coordinate System

2.3 The Item

QGraphicsItem is the base class for graphical items in a scene. Graphics View provides several standard items for typical shapes, such as rectangles (QGraphicsRectItem), ellipses (QGraphicsEllipseItem) and text items (QGraphicsTextItem), but the most powerful QGraphicsItem features are available when you write a custom item. Among other things, QGraphicsItem supports the following features:

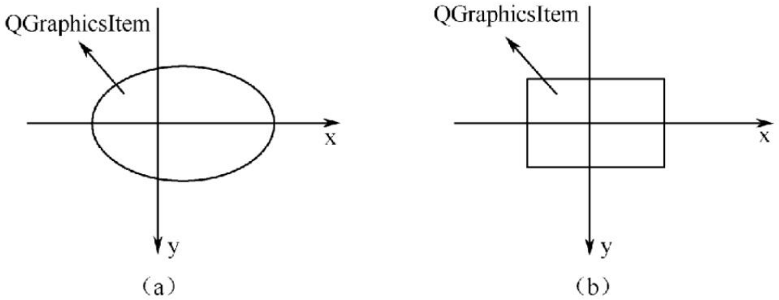
Mouse press, move, release and double click events, as well as mouse hover events, wheel events, and context menu events.

Keyboard input focus, and key events
Drag and drop
Grouping, both through parent-child relationships, and with QGraphicsItemGroup
Collision detection


Qt
A rich text item with hyperlinks, drag & drop and full text editor interaction!
Qt

Basic primitives in the graphics view frame

Primitives use their own local coordinates, and this coordinate system usually takes the center of the primitive as the origin, which is also the origin of all transformations. The coordinate direction of the primitive is the positive direction of the X axis to the right, and the positive direction of the Y axis is downward.



Item Coordinate System

3. Graphic View Framework Application

3.1 Application of Robot Lidar SLAM Mapping

At present, the most advanced navigation method used by AGV manufacturers is the positioning and navigation technology of lidar + SLAM.

SLAM allows the mobile robot to position itself according to the sensors it carries and the perception of the surrounding environment during the movement process without the information of the surrounding environment, and at the same time incrementally build a map of the surrounding environment. SLAM can improve the autonomous ability and environmental adaptability of mobile robots, and realize autonomous positioning and navigation in unknown environments. SLAM is applied to the AGV logistics trolley, without pre-laying any tracks, which is convenient for the upgrade of the factory production line and the change of the navigation route, real-time obstacle avoidance, strong environmental adaptability, and better coordinated control of multiple AGV trolleys.

The graphical view framework visualizes the detailed map in the process of robot mapping and scanning. Nodes can be edited, which simplifies the use process, reduces the use threshold, and improves efficiency.

SLAM Mapping Example

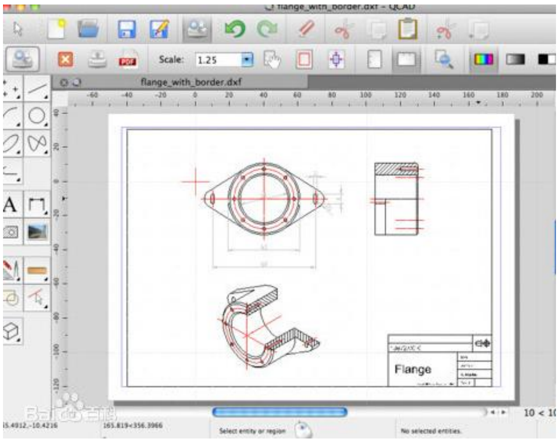
3.2 Lightweight CAD vector drawing software

Support drawing primitives: lines, ellipses, polygons, polylines, Bezier curves, etc.

Support copy, paste, cut, delete, zoom in, zoom out, undo and redo; zoom function follows mouse zoom.

Support rotation, mirroring, offset and other operations.

Alignment, equal spacing, equal width and equal height operations; grouping and ungrouping operations



flange_with_border.dxf - QCAD

Scale: 1.25

Select entity or region

No selected entities.

The use of lightweight CAD in industrial automation projects

3.3 Application in office automation

Quickly draw flowcharts, mind maps, network diagrams, industrial design, and mixed graphics and text from zero basics, all in one go.

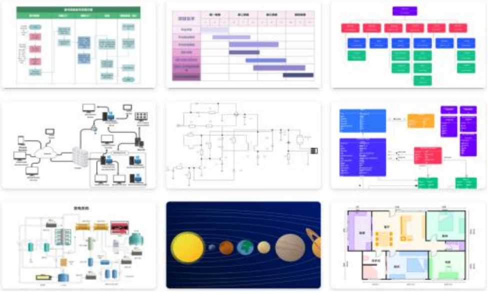


Figure Application in office automation

4. Conclusion

The graphic view framework is a very powerful and complex system, widely used and flexible. Its object-oriented programming method has strong expansibility and high reusability, and has great reference value in the process of software localization. It is used in industrial automation, robot commissioning kits and image processing. It has far-reaching significance to study the structure and application of this framework.

References

[1] Bounded model checking of C++ programs based on the Qt cross - platform framework[J]. Felipe R. Monteiro,Mário A. P. Garcia,Lucas C. Cordeiro,Eddie B. Lima Filho. Software Testing, Verification and Reliability. 2017(3)

[2] Wireless BMI System with QT Application[J]. Shreyans Shah. Networking and Communication Engineering. 2017(6)

[3] An Ameliorated Linux Based Navigational Device[J]. P. Saravanan. Programmable Device Circuits and Systems. 2012(12)

[4] Intelligent graphical user interface design utilizing multiple fuzzy agents [J] . A Agah,K Tanie. Interacting with Computers . 1999 (5)