Interactive Metro Map for Moscow and St.-Petersburg

Nikita Karpinsky, Evgeny Linsky
and Alexander Malakhov

State University of Aerospace Instrumentation
(SUAI lab)

Problem statement

- MeeGo and Symbian^1 support
- Map dragging and scaling implementation
- Pinch-to-zoom for MeeGo support
- UI is designed in QML, program logic is in C++
- Shortest route calculation
- Several maps supported

Main problem: fast map rendering


Проспект Мира
Сухаревска
елорусская
Трубная
Цветной бульвар
Тургеневская
Сретенский бульвар
Тверская
Кузнецкий мост
Чист
Чеховская
Лубянка
Пушкинская
Площадь
Революции
Арбатская
Ки
лександровский
Театральная
Охотный ряд
Боровицкая
Арбатская
Библиотека им
Ленина
инская
Новокузнецкая
Третьяковская
Парк Культуры
Полянка

Rendering map as a single image

Map is an image loaded in QML.

```
Flickable {
    function recount() {
        map.height = default_height * zoom_value;
        map.width = default_width * zoom_value;
    }
    Image {
        id: map
        source: “spb_map.jpg"
    }
}
```

Dragging is very slow.

Rendering map as multitude of images

- Images replace each other without being zoomed.

```
Flickable {
    function recount() {
        map.source = “spb_map_” + zoom_value + “.jpg”;
    }
    Image {
        id: map
        source: “spb_map_1.jpg"
    }
}
```

- Scaling is discrete.

Rendering map as visible part of vector graphic

Map is a QDeclarativeItem, only visible part is rendered.

```
void paint(QPainter *painter, /*other arguments*/) {
    QRectF rect = boundingRect ();
    for (int i = 0; i < edges.count(); ++i) {
        if (rect.contains(edges[i].x1, edges[i].y1) ||
        rect.contains(edges[i].x2, edges[i].y2))
            painter->drawLine(edges[i].x1, edges[i].y1,
                edges[i].x2, edges[i].y2);
    }
}
```

Dragging is very slow.

Creating QPixmap once for each map scale

Pixmap is created for each scale. While dragging the pixmap is drawn on the painter.

```
void createPixmap(){
    pixmap = new QPixmap(width, height);
    QPainter* painter = new QPainter(pixmap);
    //draw map
}
void paint(QPainter *painter, /*other arguments*/) {
    QRectF rect = boundingRect ();
    painter->drawPixmap(rect, *pixmap, rect);
}
```

Dragging is fast.

Creating pixmap using Qt SVG

Instead of using drawing functions an SVG image is used.

```
void createPixmap(){
    pixmap = new QPixmap(width, height);
    QPainter* painter = new QPainter(pixmap);
    QGraphicsSvgItem* svg = new QGraphicsSvgItem(“path”);
    svg->paint(painter, option);
}
```

It works more slowly than previous method.

Rendering times

(X + Y) means rendering times in QML + rendering times in C++.
Times for QML are very inaccurate.

Method	Dragging	Scaling
Single image	(10+0)ms	(10+0)ms
Multitude of images	(4+0)ms	(6+0)ms
Visible part is rendered in C++	(3+20)ms	(3+20)ms
Creating pixmap for each scale	(3+0.5)ms	(3+60)ms
Using Qt SVG	(3+0.5)ms	(3+230)ms