

УДК 004

П.А. Пех, А.О. Серета, В.О. Плотніков,  
Луцький національний технічний університет

## ПОРІВНЯННЯ ТЕХНОЛОГІЙ ПРОГРАМУВАННЯ ОБ'ЄКТІВ 2D-ГРАФІКИ ЗАСОБАМИ FLASH ТА В СЕРЕДОВИЩІ Qt НА БАЗІ C++

*У роботі зроблено спробу порівняти технології створення графічних об'єктів засобами Flash і в середовищі Qt на базі C++.*

*Ключові слова: програмування, Flash, середовище Qt на базі C++.*

*Рис. 2. Літ. 2.*

**Постановка проблеми.** Flash – це мультимедійна платформа компанії Adobe для створення додатків та мультимедійних презентацій. Використовуючи мову програмування ActionScript, Flash несе в собі не тільки великі можливості для створення презентацій, а й являє собою потужний двигун для створення браузерних ігор. У теперішній час популярність Flash зменшується, і в той же час спостерігається зростання популярності інших платформ. Аналізуючи статистику, можна віддати перевагу мові C++. Середовищ для розробки на C++ значно більше, ніж на Flash, і серед них варто зупинитися на такому середовищі, як Qt [1,2]. Великою перевагою цього середовища є те, що на Qt можна писати на будь-якій платформі і під будь-яку платформу. У Qt5 навіть з'явилась можливість писати для Android, що відкриває перед Qt нові перспективи. Виникає питання, якою мірою середовище Qt на базі C++ є конкурентним по відношенню до Flash з точки зору програмування об'єктів 2D-графіки. Ця проблема досліджується у даній статті.

**Аналіз останніх досліджень та публікацій.** Питанням програмування об'єктів 2D-графіки як засобами Flash, так і засобами Qt присвячено багато праць [1,2], однак у цих роботах не приділялося достатньо уваги їх порівняльному аналізу.

**Невирішені частини проблеми.** В даній роботі здійснена спроба порівняння технологій програмування об'єктів 2D-графіки засобами Flash та Qt. На наш погляд, ця частина проблеми недостатньо досліджена як вітчизняними, так і зарубіжними вченими, а тому її вирішення має певне теоретичне та прикладне значення.

**Основні результати дослідження.** З метою дослідження сформульованої вище проблеми, нами розроблена програма, що моделює ефект руху кольорових світлячків за курсором миші, як засобами Flash, так і засобами Qt. Оцінювався обсяг стрічок програмного коду, час, витрачений на складання програм, та інші чинники.

Оскільки Flash - це мультимедійна платформа, то робота з графікою є основою, завдяки якій і створюються Flash-додатки. Основними класами Flash для роботи з 2D-графікою, є класи MovieClip та Sprite. MovieClip - це клас для створення графічних об'єктів, який має часову шкалу, тобто може змінюватись з часом, створюючи анімацію. Сам додаток створюється як екземпляр класу MovieClip. Що стосується Sprite, то це клас для створення звичайних графічних об'єктів.

Використовуючи інструменти Flash, для реалізації задачі було написано два класи: Particle та ParticleBox.

Particle – клас, який відповідає за одну частинку і унасліджується від класу Sprite.

```
package {
    import flash.display.Sprite;
    public class Particle extends Sprite {
        public function Particle(color:uint=0xFFFFFFFF) {
            graphics.beginFill(color);
            graphics.drawRect(-5,0,10,1);
            graphics.endFill();
        }
        public var speed:Number;
        public var rotationspeed:Number;
    }
}
```

У складі класу Particle використано один метод – конструктор. Цей метод, використовуючи можливості Sprite, малює одну частинку заданого кольору. У складі класу Particle також використано дві функції – методи, які задають швидкість руху частинки і швидкість її повороту.

ParticleBox – клас, який містить у собі певну кількість цих частинок і організовує їх створення та рух.

```

package {
import flash.display.Sprite;
import flash.events.Event;
import flash.geom.Point;

public class ParticleBox extends Sprite {
private var pt:Particle;
private var particles:Array = [];

public function ParticleBox(num:Number = 100, color:uint =
0xFFFFFFFF) {
createParticles(num, color);
addEventListener(Event.ENTER_FRAME, moveparticle);
}
private function moveparticle(e:Event):void
{
for each(var item:Particle in particles)
{
var point:Point = item.globalToLocal(new
Point(stage.mouseX,
stage.mouseY));
if(point.y >= 0) {
item.rotation += item.rotationspeed;
}
else
{
item.rotation -= item.rotationspeed;
}
item.x += Math.cos(item.rotation * Math.PI / 180) *
item.speed;
item.y += Math.sin(item.rotation * Math.PI / 180) *
item.speed;
}
}
private function createParticles(num:Number, color:uint)
{
for(var i:int = 0;i < num; i++)
{
pt = new Particle(color);
pt.rotation = Math.random() * 360;
pt.speed = Math.random() * 5 + 5;
pt.rotationspeed = Math.random() * 3 + 5;
pt.x = Math.random() * 800;
pt.y = Math.random() * 600;
addChild(pt);
particles.push(pt);
}
}
}
}

```

У цьому класі вже описано три методи:

- ParticleBox – конструктор, який створює певну кількість частинок та задає функцію на подію зміни кадру;
- moveparticle – функція, яка виконується при зміні кадру та відповідає за рух частинок відносно позиції курсору миші;

- createParticles – функція, яка відповідає за створення частинок з відповідними параметрами, відображення їх на екрані та збереження для подальшої взаємодії з ними.
- На рис.1 наведено фрагмент анімації з демонстрації ефекту руху кольорових світлячків за курсором миші.

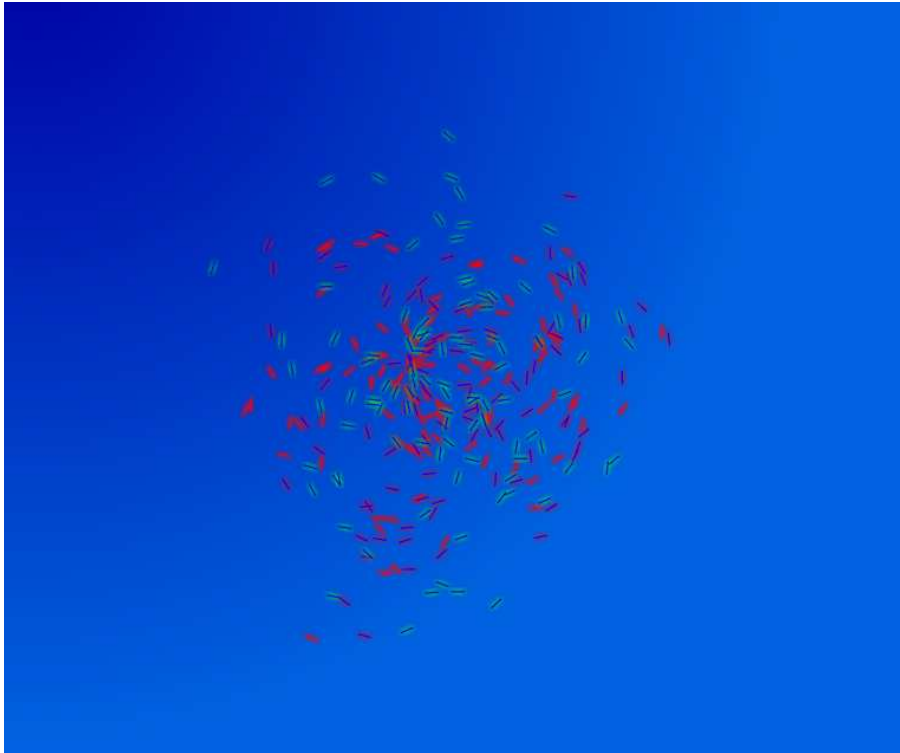


Рис. 1. Фрагмент анімації з демонстрації ефекту руху кольорових світлячків за курсором миші

Розглянемо реалізацію сформульованої вище задачі демонстрації руху за мишею кольорових світлячків, але в середовищі Qt на базі C++.

В Qt, як і в Flash, кожна програма являється екземпляром певного класу. Якщо у Flash це був MovieClip, то в Qt ми зупинилися на класі QWidget, який по-суті являється простим "контейнером" для об'єктів.

Проект складається з основного файлу main.cpp, який в однойменній функції показує наш віджет та повертає код завершення програми.

```
#include "particles.h"
#include <QApplication>

int main(int argc, char *argv[]) {
    QApplication a(argc, argv);
    Particles w;
    w.show();

    return a.exec();
}
```

Всі прототипи методів та властивості основного класу оголошуються у файлі particles.h.

```
#ifndef PARTICLES_H
#define PARTICLES_H
#include <QWidget>
namespace Ui {
    class Particles;
}
```

```

class Particles : public QWidget
{ Q_OBJECT

public:
    explicit Particles(QWidget *parent = 0);
    ~Particles();

public slots:
    void move();

protected:
    void paintEvent(QPaintEvent *event);

private:
    Ui::Particles *ui;
    float x[1001], y[1001], rotation[1001];
};
#endif // PARTICLES_H

```

При оголошенні класу `Particles` використане таке ключове слово, як `slots`. `Slot` – це аналог `EventListener`'а у `Flash`. Тобто, він спрацьовує за певної події (`SIGNAL`). Оскільки у `Qt` немає навіть такого поняття, як кадр, то події, яка спрацьовує під час його зміни, також немає. Отже, нам довелося ініціалізувати таймер, приклад якого доступніше розкриває суть сигналів та слотів.

```

QTimer *timer = new QTimer(this);
connect(timer, SIGNAL(timeout()), SLOT(move()));
timer->start(30);

```

Тобто, коли пройде заданий час, буде викликана функція `move`, реалізація якої знаходиться у файлі `particles.cpp`. Саме тут знаходяться тіла функцій класу. Починається він, як і всі файли, з директив компілятора.

```

#include "particles.h"
#include "ui_particles.h"
#include "QPainter"
#include "QTimer"
#include "QRect"
#include "QPoint"
#include "QBrush"
#include "QCursor"
#include "math.h"

```

Як засіб для малювання ми використали клас `QPainter` з достатньо широкими можливостями. Він малює на всіх об'єктах, які унаслідують від класу `QPaintDevice`, а серед цих об'єктів є і `QWidget`. Програма створення частинки у `Qt` буде виглядати наступним чином.

```

int x = int(qrand()) % 400;
int y = int(qrand()) % 400;
int rotation = int(qrand()) % 360 - 180;
QPainter paint;
paint.begin(this);
paint.setRenderHint(QPainter::Antialiasing);
paint.rotate(rotation);
paint.setBrush(QBrush(Qt::red));
paint.drawRect(x, y, 6, 2);

```

Варто звернути увагу на стрічку `paint.setRenderHint(QPainter::Antialiasing)`, яка засвідчує, що `QPainter` підтримує згладжування. Виглядає це досить непогано навіть у порівнянні з `OpenGL`. На рис.2 наведено приклад, який це демонструє.

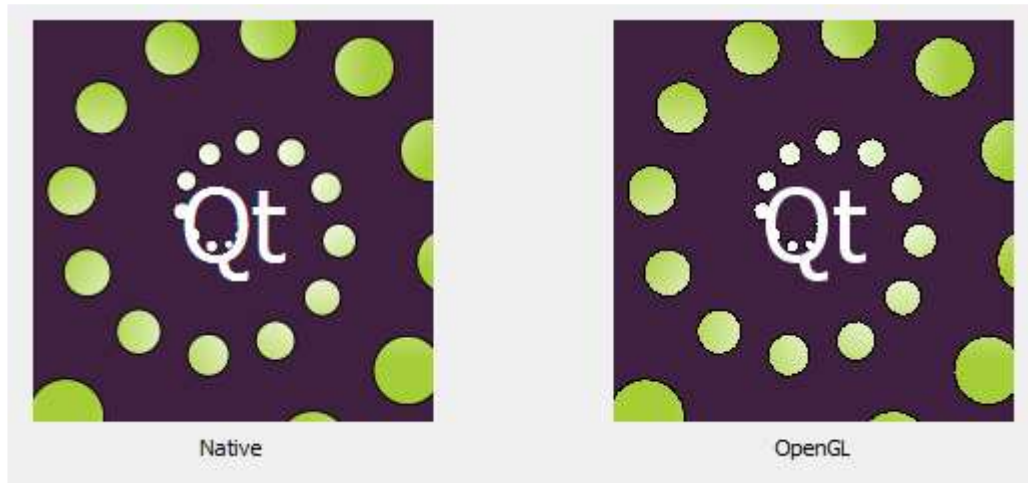


Рис.2. Демонстрація ефекту згладжування за допомогою `paint.setRenderHint(QPainter::Antialiasing)`

**Висновки.** Середовище Qt на базі C++ є достойним конкурентом Flash, адже засобами Qt можна порівняно просто реалізувати однакові можливості, використовуючи майже рівну кількість стрічок програмного коду. А прогресивний розвиток та кросплатформенність тільки додають великий плюс у сторону використання Qt.

1. Жасмин Бланшетт, Марк Саммерфилд. Qt4: Программирование GUI на C++. узд. 2-е(+CD). – М: «Кудиц-Пресс», 2008. -736 с.
2. Макс Шлее. Qt4:5. Проффессиональное программирование на C++. –СП: «БХВ-Петербург», 2009. -896