## Санкт-Петербургский государственный политехнический университет

Факультет технической кибернетики

Кафедра компьютерных систем и программных технологий

# Курсовая работа по курсу «Программирование»

# Тема работы: «Разработка приложения с графическим интерфейсом на языке C++ с использованием библиотеки QT»

Выполнил: Мурашко Денис, группа 1081/3

Преподаватель: доц. Пышкин Е.В.

## Техническое задание

Разработать игру между человеком и компьютером. Имеются танк-игрок и танки-боты, которые двигаются по игровому полю (танк-игрок с клавиатуры, танк-бот самостоятельно). Танки могут стрелять пулями. Задача попасть пулей в танк (убийство). Выиграет тот, кто убьет противников.

Логической составляющей проекта будет угол движения (танк может стрелять только в ту сторону, в которую движется) и геометрия, связанная с попаданием пули.

Данное приложение разработать на языке C++ с использованием библиотеки Qt для реализации графического интерфейса пользователя.

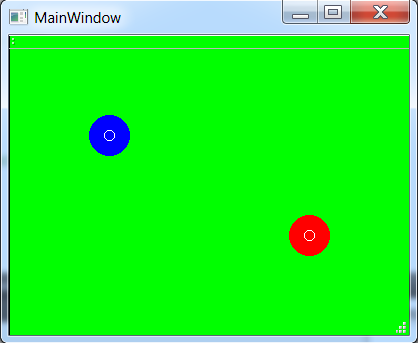
Требования к графическому интерфейсу

Танк-игрок находится в какой-нибудь точке игрового поля. Поле для удобства занимает один экран. Танк имеет форму круга - для облегчения поворотов. Так же он имеет башню, которая направлена в сторону его движения. Пули танк выпускает в форме отрезка или кружка (неважно), которые вылетают в сторону движения танка и траектория которых не зависит от дальнейшего движения танка. Движение танка-игрока описывается клавишами ‘w’ , ’s’ , ’d’ , ’a’ –прямо, назад, вправо и влево. Танк-игрок выпускает пулю на ‘space’. Также должно уведомляться об итоге игры (победа/поражение).

## Образ и границы проекта

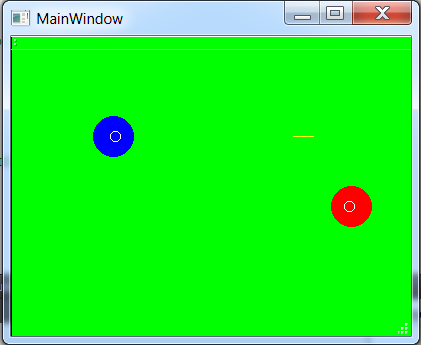
Выглядеть примерно будет так:

Начало игры



Игра

Из скриншота видно, в какую сторону движутся танки (башня немного смещена в сторону движения) и что танк стреляет в сторону своего движения.



Конец игры

Убийство танка – попадание пули в танк.(еще видно что траектория пули не зависит от направления движения танка(танк направлен в другую сторону после выстрела)).





## 2. Проектирование системной архитектуры

Классы отвечающие за логику игры

Танк-игрок

struct TankCoordinates//координаты танка-игрока

{

int x;

int y;

};

struct TankVec//для направления движения танка

{

int x;

int y;

};

class Igrok//танк-игрок

{

public:

TankCoordinates koords;

TankVec vec;

Field field;

QVector <Bullet> bullet;//вектор пуль

int r;//радиус танка

int speed;//скорость движения танка

Igrok();//контструктор

//void draw(QPainter &painter);//рисование танка-игрока

};

Танк-бот

struct BotCoordinates //координаты

{

int x;

int y;

};

struct VecBot //угол движения

{

int x;

int y;

};

class Bot

{

public:

BotCoordinates koordsb;

VecBot vecb;

Field field;

QVector <Bullet> bbullet;//вектор пуль

int alpha; //угол движения

int speedb; //скорость дижения

int r;//радиус бота

Bot();//конструктор

//void draw(QPainter &painter);//рисование бота

};

Пуля

struct BulletCoordinates//координаты пули

{

int x;

int y;

};

struct AlphaMove//угол движения пули

{

int x;

int y;

};

class Bullet:public QObject //класс пуля

{

Q\_OBJECT

public:

BulletCoordinates coords;

AlphaMove alpha;

int speed;//скорость пули

int l;//длина пули

Bullet();//контсруктор

Bullet(const Bullet&);//компилятор потребовал написания этой функции

Bullet& operator =(const Bullet&);//компилятор потребовал написания этой функции

void drawp(QPainter &painter);//рисование пули

};

Игровое поле

struct Field //игровое поле

{

//границы поля

int x1;

int y1;

int x2;

int y2;

Field();//конструктор

};

Классы графического интерфейса

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

void *paintEvent*(QPaintEvent \*event);

~*MainWindow*();

void proverkaDeath();//проверка убийства

bool proverkaFieldIgrok();//проверка границ поля для игрока

bool proverkaFieldBot();//проверка границ поля для бота

void drawIgrok(QPainter &painter);//рисование танка-игрока

void drawBot(QPainter &painter);//рисование бота

public slots:

void moveBulletOnTimeout();//движение пули

void driveBotOnTimeout();//движение бота

void hitBulletBotOnTimeout();//выпуск пули бота

void moveBotBulletOnTimeout();//движение пули бота

private:

Ui::MainWindow \*ui;

void *keyPressEvent*(QKeyEvent \* ev);

QPainter painter;

Igrok tank;

Bot bot;

//таймер для

QTimer\* BulletTimer; //движения пули игрока

QTimer\* BotTimer;//движения бота

QTimer\* BulletBotTimer;//выпуск пуль у бота

QTimer\* BBulletTimer;//движение пуль у бота

};

Тестовый класс

class Test

{

Igrok t;

Bot bott;

bool proverkaFieldIgrok();

bool proverkaFieldBot();

};

## 3. Требования к используемым библиотекам графического интерфейса

#include <QMainWindow>

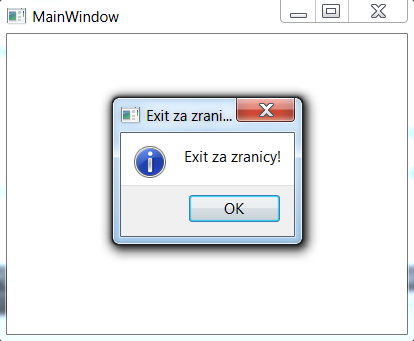
#include <QtGui>

#include <QPainter>

#include <QTimer>

## 4. Обработка ошибок

## Танк-игрок и танк-бот не могут заезжать за границы игрового поля. В процессе игры эти ошибки исправляются, т.е. танк-игрок стоит на месте , а танк-бот движется в другую сторону. Если неправильно заданы начальные координаты игрока или бота, то выводится сообщение об ошибке.



5.Тестирование приложения

Процесс тестирование приложения происходит в основном процессе игры (видно наглядно).Видно, что танки стреляют только в сторону своего движения, танк-игрок движется и стреляет с помощью клавиатуры, а танк-игрок самостоятельно. В процессе тестирования в логике игры проблем не обнаружено, но возникают иногда некоторые проблемы с графическим интерфейсом:

* танк-бот зависает на некоторое время, а потом движется “бабочкой” как бы догоняя упущенное время
* с зависанием связано и появление “двойных пуль” (видно на четвертой картинке)

- танк игрок бывает некоторое время не может двигаться

- в процессе длительной игры пули начинают ускоряться (я думаю что связано с таймером, т.к. скорость пули нигде не меняется).

Так же игра идет только с одним танком-ботом (не реализовано).