Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

\_\_\_\_\_\_\_Програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЕКТ**

з \_\_Об’єктно-орієнтованого програмування\_\_

(назва дисципліни)

на тему:\_\_\_**Створення моделі акваріума засобами мови С++**\_\_

Виконав: студент (ка) \_\_1\_\_\_ курсу, групи\_\_\_143\_\_\_\_

спеціальності \_\_\_\_\_6.121\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр спеціальності)

\_\_\_\_Інженерія програмного забезпечення\_\_\_\_\_\_\_\_

(назва спеціальності)

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Мовченюк Р.В.\_\_\_\_\_\_

(підпис) (прізвище, ініціали)

Керівник \_\_\_\_\_\_\_Миронів І.В.\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

Нормоконтролер \_\_\_\_\_\_\_Комісарчук В.В.\_\_\_\_\_\_\_\_

(прізвище, ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| **До захисту допущено:** |  |  | **Дата захисту** «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |
| від «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |  |  | **Оцінка:** |
| Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | за національною шкалою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (словами) |
| Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (цифра) |
|  |  |  | за шкалою ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  | (літера) |

Чернівці–20\_17\_

Чернівецький національний університет імені Юрія Федьковича

Кафедра \_Програмного забезпечення комп’ютерних систем\_\_\_\_

Спеціальність \_Інженерія програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсовий проект студента**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Мовченюк Роман Валерійович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема проекту \_**Створення моделі акваріума засобами мови С++**\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Термін подачі студентом завершеного проекту «\_\_1\_» \_\_червня\_\_\_ 20\_17\_ р.

3. Вихідні дані до проекту:

­­­­­­– визначити об’єкт TFish – акваріумна рибка;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

– визначити об’єкти ТPike і TKarp, які є наслідниками об’єкта TFish;\_\_\_\_\_\_\_\_\_

– визначити об’єкт TAquarium, звязні списки об’єктів TPike і TKarp є полями об’єкта TAquarium;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

– дозволити щукам поїдати коропів як тільки вони їх побачать, і коропам втікати від щук.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити):

– розробити загальні вимоги до програми;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

– описати модулі програми та алгоритми роботи у вигляді блок-схем;\_\_\_\_\_\_\_\_\_

– описати методи програми;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

– опис користувацького інтерфейсу;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

– зробити висновки по розробленій програмі;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

– навести текст програми у вигляді програмного коду.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

– блок-схеми роботи з програмою;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

– скріншоти роботи з програмою.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання прийняв (ла) до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Керівник проекту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**РЕФЕРАТ**

В курсовому проекті розроблено графічну модель акваріума.

Інтерфейс програми розрахований на користувачів, які не мають спеціальної комп’ютерної підготовки. Досвід роботи не вимагається.

Розробка реалізована засобами фреймворка Qt та мови С++. Дане середовище є зручним у використанні для відображення та реалізації найпростішого руху графічних об’єктів.

Дана розробка у майбутньому може бути розширена із добавленням нового функціоналу і видозміненою логікою обробки.

Курсовий проект містить: 20 с., 3 рис., 3 табл., 1 додаток, 2 джерела.

*ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, АКВАРІУМ, ПЕРСОНАЛЬНІ КОМП’ЮТЕРИ, СЕРЕДОВИЩЕ РОЗРОБКИ QT 4.8,МОВА ПРОГРАМУВАННЯ С++.*

**ЗМІСТ**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ……………………………………….........-5-

[1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ - 6 -](#_Toc484504667)

[1.1. Загальні вимоги до програми - 6 -](#_Toc484504668)

[*1.1.1. Вимоги до графічного інтерфейсу користувача* - 6 -](#_Toc484504669)

[*1.1.2. Вимоги до архітектури програми* - 6 -](#_Toc484504670)

[*1.1.3. Вимоги до функціональності додатка* - 6 -](#_Toc484504671)

[1.2. Призначення та область застосування - 7 -](#_Toc484504672)

[1.3. Функціональні вимоги - 7 -](#_Toc484504673)

[2. ОПИС ПРОГРАМИ - 9 -](#_Toc484504674)

[2.1. Структура програми - 9 -](#_Toc484504675)

[*2.1.1. Модулі програми* - 9 -](#_Toc484504676)

[*2.1.2.* *Алгоритми роботи програми* - 9 -](#_Toc484504677)

[2.2. Опис методів програми - 10 -](#_Toc484504678)

[2.3. Програмні засоби - 13 -](#_Toc484504679)

[2.4. Опис користувацького інтерфейсу - 13 -](#_Toc484504680)

ВИСНОВОК……………………………………………………………………..- 16 - СПИСОК ВИКОРИСТОНОЇ ЛІТЕРАТУРИ…………………………………..- 17 -

ДОДАТКИ……………………………………………………………………….- 18 -

Додаток А. Скролінг (текст) програми…………………………………...- 18 -

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ**

ООП – об’єктно-орієнтоване програмування.

Фреймворк (платформа, структура) – інфраструктура програмних рішень, що полегшує розробку складних програмних систем.

Qt – інструмент розробки ПО на мові програмування С++.

Слот – спеціальні методи фреймворку Qt, які викликаються сигналами

Сигнал – генератори виклику слотів в Qt.

# АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ

## Загальні вимоги до програми

### Вимоги до графічного інтерфейсу користувача

### Робоча мова інтерфейсу англійська.

### Кнопок керування повинно бути п’ять: додати коропа, додати щуку, почати рух, зупинити рух, очистити акваріум.

### Кольорова гама (палітра): фон – темно-синій, з фоновим малюнком акваріуму, короп – зелений трикутник, щука – червоний трикутник.

### Вимоги до архітектури програми

1. Використання стандартних контейнерних класів які реалізують**:**

* додавання риб до акваріуму;
* малювання елементів акваріуму і риб на графічній сцені;
* методи для руху об’єктів;
* вилучення риб з акваріуму;
* оцінювання поведінки риби в залежності від того, що вона «бачить» перед собою.

### Вимоги до функціональності додатка

1. Запуск графічного режиму, промальовування фонового малюнку акваріуму. Відображення функціоналу програми.
2. Після натиску користувача на кнопку “Add pike” або “Add karp”, програма почне опрацьовувати натискання на графічну сцену, і додаватиме відповідну рибу на акваріум в місце де знаходиться курсор, якщо не перевищено ліміт на кількість риб в акваріумі (3 щуки, 10 коропів).
3. Натискання клавіші “Start” запускає таймер, який викликає через певні проміжки часу в кожної риби метод, що відповідає за рух певного типу риби.
4. Метод руху для коропа перевіряє певну ділянку акваріуму перед рибою, якщо це противник короп втікає від нього, якщо риба короп або межа акваріуму – намагається уникнути зіткнення.
5. Метод руху для щуки аналогічний за виключення того, що коли щука бачить противника вона намагається його наздогнати. Якщо полювання виявилось успішним жертва прибирається з акваріуму, колір щуки змінюється, і вона не може полювати деякий час.

## Призначення та область застосування

***Мета роботи*** полягає у розробці моделі акваріуму засобами мови С++, закріпленні знань з курсу ООП.

***Область застосування* –** програмне забезпечення для персональних комп’ютерів, може бути використано в ігрових та повчальних цілях.

## Функціональні вимоги

До програмного забезпечення висуваються такі функціональні вимоги:

1. Програмний інтерфейс повинен надавати наступні можливості:

* коропів та щук натисканням лівої клавіші миші, в місці де знаходиться курсор;
* пуску і зупинки руху риб;
* очищення акваріуму від риб.

1. Короп повинен рухати за наступними законами:

* якщо перед коропом з’явилась границя акваріуму він змінює свій напрям руху і на час повороту зменшує швидкість так щоб уникнути вильоту за межі акваріуму;
* якщо перед короп з’явився інший короп, він незначною мірою змінює напрям свого руху, намагаючись уникнути зіткнення;
* якщо короп бачить хижака він змінює свій напрям руху на протилежний і збільшує швидкість на певний час.

1. Щука повинна рухатись за наступними законами:

* якщо перед щукою з’явилась границя акваріуму вона змінює свій напрям руху і на час повороту зменшує швидкість так щоб уникнути вильоту за межі акваріуму;
* якщо перед щукою з’явилась інша щука, вона незначною мірою змінює напрям свого руху, намагаючись уникнути зіткнення;
* якщо в поле видимості щуки потрапив короп вона збільшує швидкість і змінює напрям руху в напрямку жертви до тих пір поки жертва не зникне з поля зору або не відбудеться зіткнення з нею;
* при зіткненні з коропом, короп прибирається з акваріуму, колір щуки змінюється на темніший, і вона втрачає можливість полювати на двадцять п’ять секунд, по закінченню цього часу колір щуки повертається в попереднє значення;
* максимальна кількість щук в акваріумі – 3;
* максимальна кількість коропів – 10;
* програма продовжує роботу до закриття її користувачем.

# ОПИС ПРОГРАМИ

## Структура програми

### Модулі програми

Робота розробленого програмного забезпечення реалізується наступними модулями:

1. TFish – батьківський клас, в якому реалізовані спільні методи коропів і щук.
2. TPike – клас, нащадок TPike, в якому додані методи малювання щуки, і можливість з’їдання коропа.
3. TKarp – клас, нащадок TFish, в якому додані методи малювання коропа, і можливість коропа втікати від щуки.
4. TAquarium – клас який відповідає за промальовування акваріуму, полями якого є списки об’єктів TPike і TKarp.

### Алгоритми роботи програми

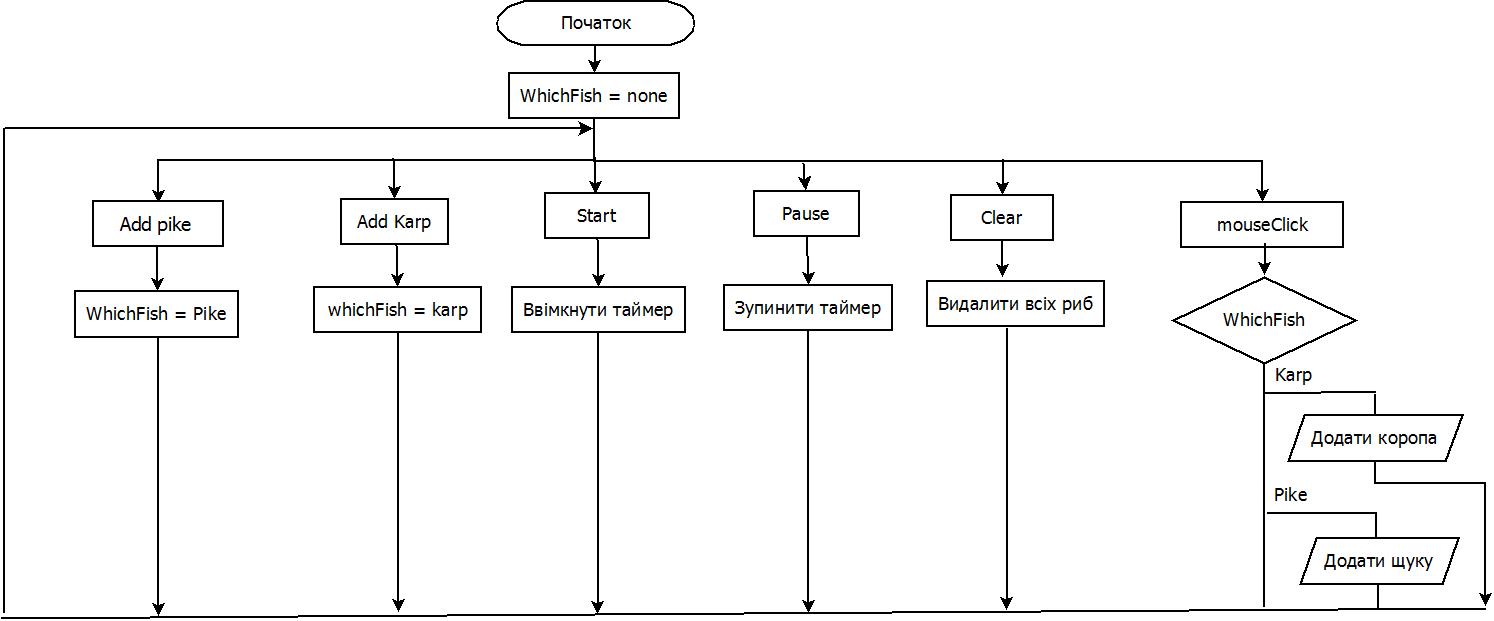


Рис. 1 – Основний алгоритм програми

## Опис методів програми

Список методів класу *TAquarium* та їх опис наведено в табл.1.

Таблиця 1 – Основні методи класу TAquarium

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1 | *TAquarium()* | Конструктор, виділяє пам’ять для графічної сцени, задає їй початкові налаштування, промалювує фон і межі акваріума |
| 2 | *mousePressEvent(QMouseEvent\*)* | Опрацьовує натискання клавіші миші на акваріум. І відносно того яка клавіша останньою була натиснутою додає на сцену і в список коропа чи щуку |
| 3 | *on\_start\_clicked()* | Запускає таймер, який викликає методи руху усіх риб |
| 4 | *on\_addPike\_clicked()* | Сигналізує про те що при натисканні мишею на акваріум потрібно додавати щуку |
| 5 | *on\_addKarp\_clicked()* | Сигналізує про те що при натисканні мишею на екран потрібно додавати коропа |
| 6 | *on\_clear\_clicked()* | Очищає акваріум і динамічні списки риб від усіх риб |
| 7 | *on\_pause\_clicked()* | Зупиняє таймер |

Список методів класу *TFish* та їх опис наведено в табл.2.

Клас *TFish* є наслідником стандартного класу фреймворку Qt – *QGraphicsItem*, в якому присутні два віртуальні методи *boundingRect(),* який визначає прямокутну область в які знаходиться об’єкт(в даному випадку це червоний або зелений трикутник), та метод *paint* який малює об’єкт в визначеній вище області.

Таблиця 2 – Основні методи класу TFish

|  |  |  |
| --- | --- | --- |
| № | Методи | Короткий опис |
| 1 | *lookLandSpace(QList<QGraphicsLineItem\*>)* | Перевіряє чи риба бачить перед собою елементи ландшафту, якщо так, то змінює відповідним чином напрям руху і швидкість |
| 2 | *lookFriend(QList<QGraphicsItem\*>)* | Перевіряє чи бачить риба рибу свого типу(коро коропа, щука щуку), якщо так, то змінює напрям і швидкість відповідним чином, щоб уникнути зіткнення. |
| 3 | *priorityMove()* | Допоміжна функція яка допомагає плавно уникати зіткнень |
| 4 | *randomMove()* | Якщо риба нічого не бачить перед собою, ця функція генерує псевдорандомні напрям руху і швидкість(в допустимих межах) |
| 5 | *normalizeAngle(qreal)* | Приводить значення кута в межі від 0 до 360 градусів |
| 6 | *look()* | Метод що викликаэ методи *priorityMove*. *lookFriend*, *lookLandSpace* і визначає напрям руху риби |
| 7 | *boundingRect*() | Перегружена функція з батьківського класу *QGraphicsItem*, яка визначає межі об’єкта |

Список методів класу *TPike* наведено в табл.3.

Клас *TPike* є наслідником класу *TFish* в якому реалізовані методи руху і промальовування щуки.

Таблиця 3 – Основні методи класу TPike

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1 | *TPike(QPoint,QGraphicsScene\*)* | Конструктор, додає рибу на сцену і в динамічні списки |
| 2 | *advance(int step)* | Перевантажений метод класу *QGraphicsItem*, який викликається методом *advance()* з QGraphicsScene для всіх елементів сцени для яких він визначений, викликає функції *look()*, і змінює положення риби в акваріумі |
| 3 | *lookEat()* | Метод, що перевіряє чи бачить риба перед собою потенційну жертву, і змінює її координати в напрямку жертви, збільшуючи швидкість |
| 4 | *look()* | Перевизначений метод батьківського класу *TFish*, що викликає методи *priorityMove*, *lookLandSpace*, *lookEat*, *lookFriend* та *randomMove* в вказаному порядку якщо хоч один із них повертає значення тру функція припиняє роботу, зміниви координати і швидкість риби. |
| 5 | *eating()* | Метод який викликається методо *lookEat* коли щука наздогнала коропа.  Ця функція знищує коропа, видаливши його зі сцени і з динамічного списку, змінює колір риби, і запускає таймер на період дії якого щука не може полювати |
| 6 | *stayHungry();* | Слот, що викликається таймером що почав роботу в функції вище. Цей слот змінює колір щуки на звичайний і зупиняє таймер, що дозволяє їй полювати. |
| 7 | *paint(QPainter*\**)* | Метод який малює рибу на акваріумі |

Основні методи класу *TKarp* описані в табл.4.

Таблиця 4 – Основні методи класу TKarp

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1 | *paint(QPainter\*)* | Аналогічно до *TPike* |
| 2 | *advance()* | Аналогічно до *TPike* |
| 3 | *look()* | Аналогічно до *TPike* |
| 4 | *lookDanger()* | Перевіряє чи бачить короп перед собою щуку, якщо так змінює напрям руху на протилежний відносно ворога, і збільшує швидкість |

## Програмні засоби

Розробка реалізована засобами фреймворка Qt та мови С++. Qt – зручне і швидку середовище розробки, для реалізації двовимірної графіки, перевірки на перетин об’єктів, та моделювання найпростішого руху.

Були використані такі стандартні компоненти:

* QGraphicsScene – компонент що, являє собою по суті поле для розміщення графічних об’єктів, Він дозволяє зручно додавати, видаляти та перемальовувати графічні об’єкти.
* QGraphicsItem – компонент, який є полем QGraphicsScene, дозволяє працювати з малюнком як з окремим об’єктом, легко та зручно переводити координати рисунка з системи відліку сцени до локальної системи відліку об’єкта.

## Опис користувацького інтерфейсу

При запуску програми, користувач бачить порожній акваріум (див. рис.1). Щоб додати ту чи іншу рибу потрібно натиснути на клавіші “Add pike” або “Add karp”, і клацнути мишею по акваріуму (див. рис.2).

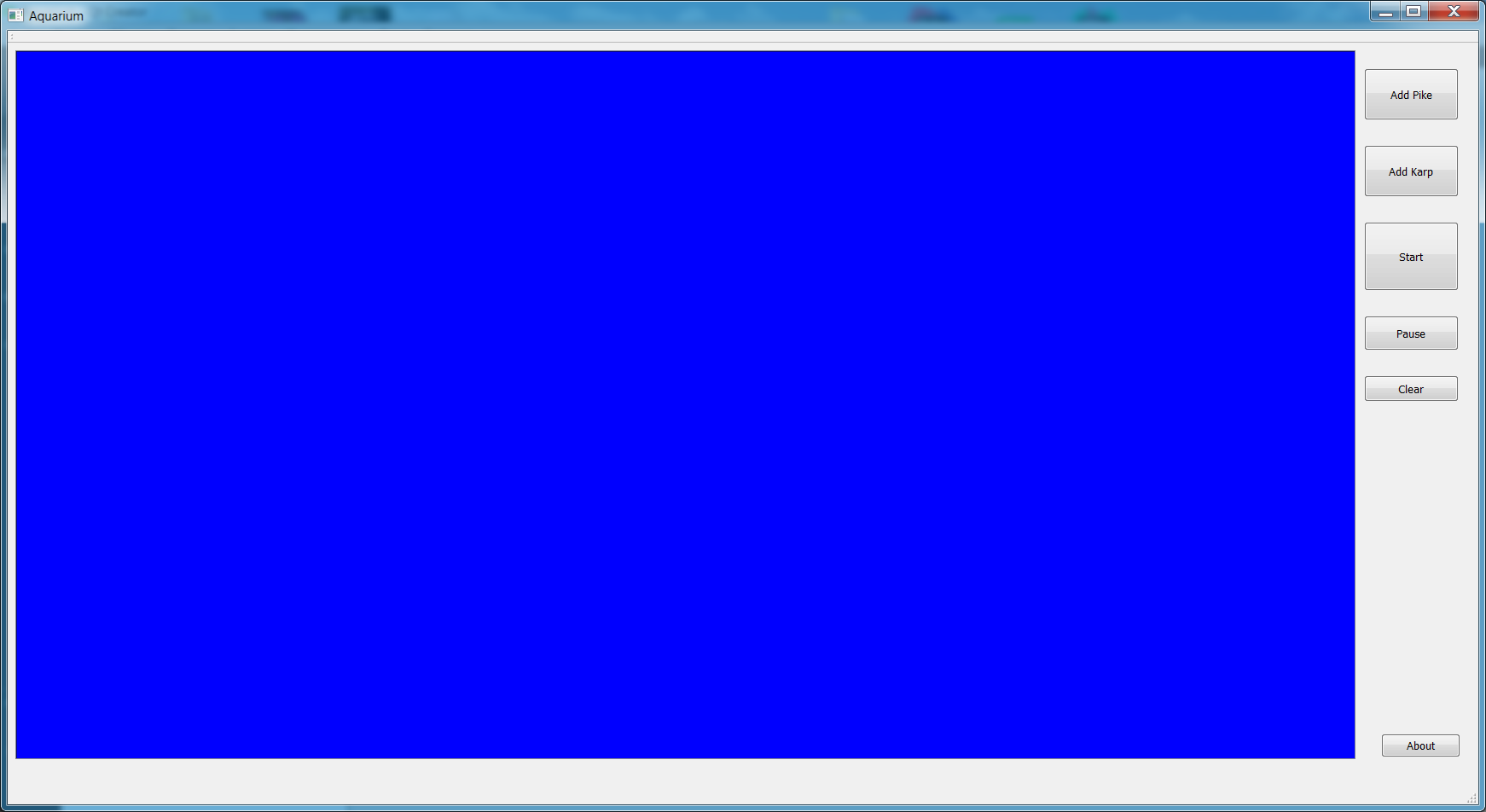


Рис. 1 – Порожній акваріум

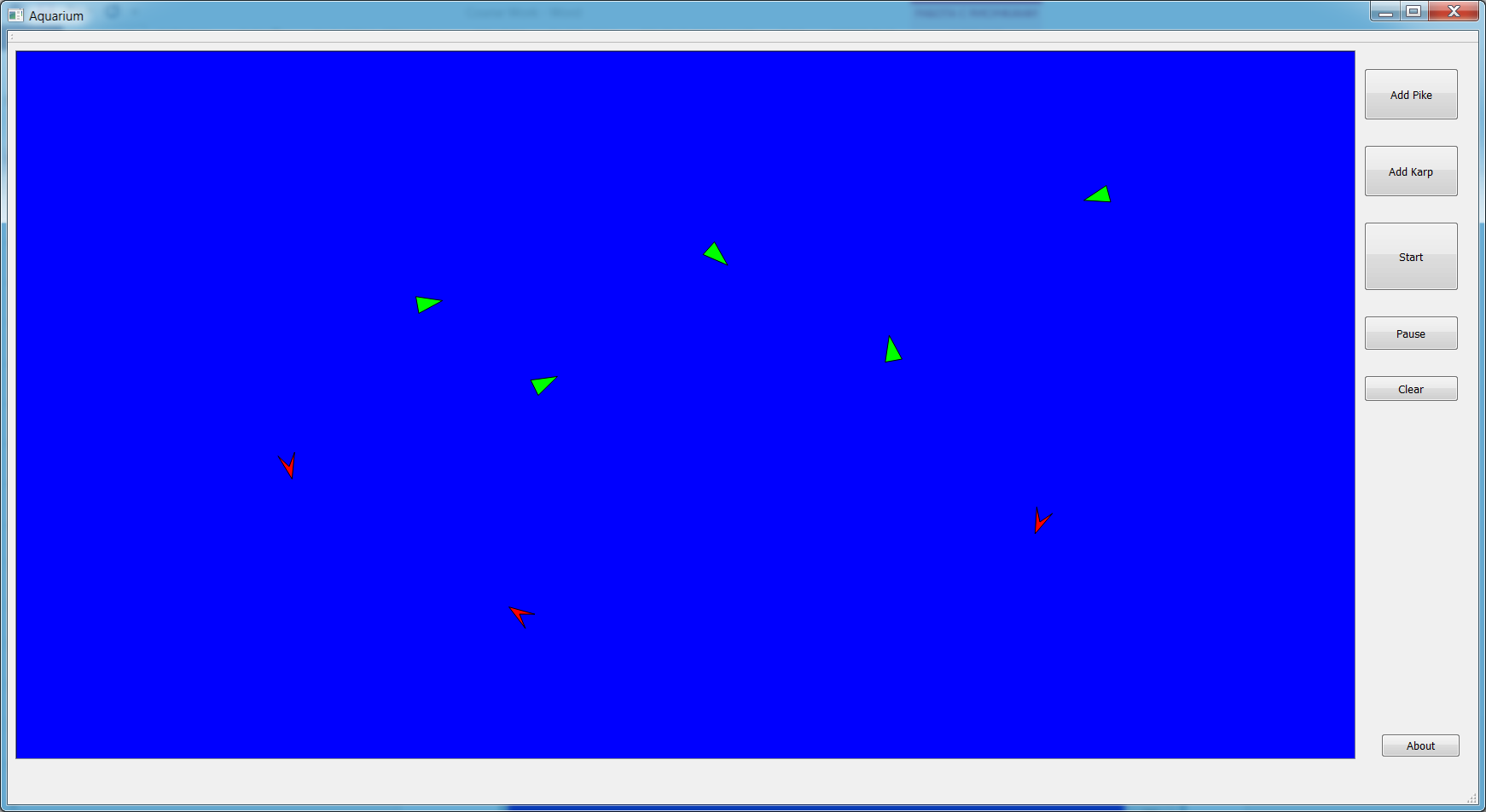


Рис. 2 – Додані риби

Щоб риби почали рухатись необхідно клацнути по кнопці “Start”, щоб принити рух – клацнути “Pause”, і щоб очистити окваріум від риб – “Clear”.

При натисканні на клавішу “About”, на екрані з’явиться інформація про розробника (див. рис.3)

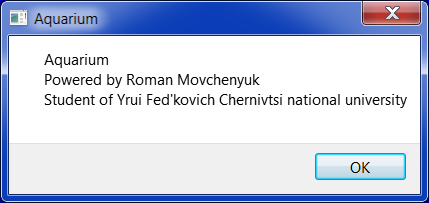


Рис. 3 – Інформація про розробника

**В**ИСНОВКИ

В результаті виконання курсового проекту розроблено модель акваріуму.

Розроблене програмне забезпечення має наступні можливості:

1. Додавати риби в акваріум.
2. Вилучати риби з акваріуму.
3. Моделювати поведінку риб.

В даній програмі використовуються засоби фреймворку Qt та об’єктно-орієнтованих можливостей мови С++.

Дана розробка у майбутньому може бути розширена із добавленням нового функціоналу і видозміненою логікою обробки.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Denis Markov [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Denis Markov © 2017. – Режим доступу:

https://www.youtube.com/user/KennyMcKormic123 (дата звернення 25.05.2017) – Назва з екрана.

1. Qt Forum [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Qt © 2017. – Режим доступу: https://forum.qt.io (дата звернення 28.05.2017) – Назва з екрана.

**ДОДАТКИ**

**Додаток А**

**Скролінг (текст) програми**

TAquarium.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QGraphicsItem>

#include <QPainter>

#include <QGraphicsScene>

#include "tkarp.h"

#include <QMouseEvent>

#include "tpike.h"

#include <QList>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

protected:

void *mousePressEvent*(QMouseEvent \*event) ; /// Перехоплення лівого кліку мишки

/// слоти що відповідають за опрацювання натискання клавіш

private slots:

void on\_start\_clicked();

void on\_addPike\_clicked();

void on\_addKarp\_clicked();

void on\_clear\_clicked();

void on\_pause\_clicked();

private:

QString whichFish; /// допоміжна змінна яка запам'ятовує яка кнопка натиснута останньою(додати коропа чи щуку)

QTimer \*timer; /// таймер що для ігрового циклу

QGraphicsScene \*scene; /// сцена

QList< QGraphicsItem\* > listOfKarp; /// список коропів

QList< QGraphicsItem\*> listOfPike; /// список щук

QList< QGraphicsLineItem\* > listOfAquarium; /// список ліній з яких складається ландшафт

void addLineToAquarium(QLineF line, QColor color); /// функція яка малює лінії на сцені

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

TFish.h

#ifndef TFISH\_H

#define TFISH\_H

#include <QObject>

#include <QGraphicsItem>

#include <QPainter>

#include <QGraphicsScene>

#include <QMainWindow>

#include<QPoint>

#include<QRectF>

#include <QGraphicsLineItem>

class TFish : public QObject,public QGraphicsItem

{

Q\_OBJECT

protected:

QRectF *boundingRect*() const override;

public:

TFish(QPointF \_point, qreal \_speed, QColor \_color, QGraphicsScene \*\_scene);

bool lookLandSpace(QList<QGraphicsLineItem\*> \*listOfAquarium); ///перегляд ландшафту перед рибою

bool lookFriend(QList<QGraphicsItem\*> \*listOfFriend);/// перегляд риб спільного типу

bool priorityMove();/// допоміжна функція повороту

void randomMove(); ///псевдорандомний рух

qreal normalizeAngle(qreal \_angle);

QPointF getPoint()const;

QColor getColor()const;

qreal getAngle()const;

void setAngle(qreal \_angle);

qreal getSpeed() const;

void setSpeed(qreal value);

void setColor(const QColor &value);

int getPriorityLeft() const;

void setPriorityLeft(int value);

int getPriorityRight() const;

void setPriorityRight(int value);

private:

int priorityLeft;

int priorityRight;

QPointF point;

qreal speed;

QColor color;

qreal angle;

protected:

QGraphicsScene \*scene;

};

#endif // TFISH\_H

TPike.h

#ifndef TPIKE\_H

#define TPIKE\_H

#include "tfish.h"

#include "tkarp.h"

#include <QTimer>

class TPike : public TFish

{

Q\_OBJECT

protected:

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

public:

TPike(QPointF \_point, QGraphicsScene \*\_scene, QList<QGraphicsItem \*> \*\_listOfKarp, QList< QGraphicsItem \*> \*\_listOfPike, QList< QGraphicsLineItem\* > \*\_listOfAquarium );

void *advance*(int step)override;

private:

QTimer \*satedPikeTimer;

void look();

bool lookEat();/// перевірка чи в полі зору жертва

void eating(QGraphicsItem \*karp);/// "з'їдання жертви"

QList< QGraphicsItem\* > \*listOfKarp;

QList< QGraphicsItem\*> \*listOfPike;

QList< QGraphicsLineItem\* > \*listOfAquarium;

private slots:

void stayHungry();

};

#endif // TPIKE\_H

TKarp.h

#ifndef TKARP\_H

#define TKARP\_H

#include "tfish.h"

#include <QList>

class TKarp : public TFish

{

Q\_OBJECT

protected:

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

public:

TKarp(QPointF \_point, QGraphicsScene \*\_scene,

QList<QGraphicsItem\*> \*\_listOfKarp,

QList<QGraphicsItem\*> \*\_listOfPike,

QList< QGraphicsLineItem\*> \*\_listOfAquarium);

void *advance*(int step) override;/// функція яка викликається таймером і відповідає за рух

void look();/// перегляд і оцінка всіх об'єктів які може бачити риба

bool lookDanger();/// перевірка чи бачить небезпеку

virtual ~*TKarp*();

private:

int prioritySpeed;

QList< QGraphicsItem\* > \*listOfKarp;

QList< QGraphicsItem \*> \*listOfPike;

QList< QGraphicsLineItem\* > \*listOfAquarium;

};

#endif // TKARP\_H