|  |
| --- |
| **Министерство образования и науки Российской Федерации** |
| Федеральное государственное бюджетное образовательное учреждение |
| высшего образования |
| **«Владимирский государственный университет** |
| **имени Александра Григорьевича и Николая Григорьевича Столетовых»** |
| **(ВлГУ)** |

**Кафедра вычислительной техники и систем управления**

**Курсовая работа**

**по дисциплине**

**«Языки программирования»**

**Демографическая модель «Зайцы и волки»**

**Выполнил:**

ст. гр. ВТ-116

Лаптев Д.С.

**Принял:**

доцент Быков В. И.

**Владимир, 2018**

Анотация

Курсовая работа по дисциплине языки программирования применяется для моделирования процесса игры. Была разработана диаграмма классов и объектов в соответствие с нормами объектно-ориентированного программирования. Так же алгоритм методов. На основе сказанного ранее написан программный код. Курсовая работа состоит из 22 листов. Работа включает в себя 4 рисунка с тестированием. В результате выполнения данной работы мы получили рабочую программу «Зайцы и волки».

Содержание

У

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ВЛГУ 09.03.01.ВТ-116.14.5.00 ПЗ

Разработал

Лаптев Д.С.

Проверил

Быков В. И.

Рецензия

Н. Контр.

Утвердил

Курсовая работа.

Пояснительная записка

Лит.

Листов

22

ВТ-116

Введение……………………………………………………………………….…...4

1 Формулировка задачи …………………………………………………………..5

2 Системный анализ …………………………………………………………..…..6

3 Архитектура программной системы ………………………………………….10

3.1 Идентификация классов и объектов ………………………………......9

3.2 Идентификация содержания классов ……………………….......…….9

3.3 Реализация классов и объектов ………………………...…………….10

4 Инструкция ……………………………………………………………….…….11

5 Программирование ………………………………………………………..……12

|  |
| --- |
|  |

6 Тестирование …………………………………………………………………....20

Заключение ……………………………………………………………………..…21

Список использованных источников …..……..……………………………..…..22

Введение

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

ВЛГУ 09.03.01.ВТ-116.14.5.00

Моим заданием является написание программы «Зайцы и волки». Для написания программы я использовал объектно-ориентированный подход. Такой подход имеет несколько преимуществ. Использование одного и того же программного кода с разными данными. Классы позволяют создавать множество объектов, каждый из которых имеет собственные значения атрибутов. Нет потребности вводить множество переменных, т.к объекты получают в свое распоряжение индивидуальные так называемые пространства имен. Пространство имен конкретного объекта формируется на основе класса, от которого он был создан, а также от всех родительских классов данного класса. Объект можно представить как некую упаковку данных. Виртуальное наследование ускоряет производительность программы, а полиморфизм позволяет не писать новый код, а настраивать уже существующий, за счет добавления и переопределения атрибутов. Это ведет к сокращению объема исходного кода.

1 Формулировка задачи

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

ВЛГУ 09.03.01.ВТ-116.14.5.00

Вариант 13. "Зайцы и волки"

На некоторой территории хаотически двигаются зайцы и волки, причем выйти за пределы этой территории они не могут. Если волк сталкивается с зайцем, то он его съедает, Заяц умирает от старости в заданном возрасте. За время жизни через равные промежутки времени каждый заяц рождает еще одного зайца, Волк умирает от старости со своей заданной продолжительностью жизни. Если волк в течение некоторого времени не съедает зайца, то он умирает от голода. Через равные промежутки времени волк рождает еще одного волка. В начальный момент времени зайцы и волки имеют случайным образом выбранный возраст. В процессе моделирования следует отображать в цифровой форме количество зайцев и их средний возраст, количество волков и их средний возраст. При инициализации процесса моделирования указываются параметры моделирования; начальное количество зайцев, начальное количество волков, продолжительность жизни зайцев, продолжительность жизни волков, интервал между рождением зайцев, интервал между рождением волков, интервал жизни волка после съедения последнего зайца.

2Системный анализ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

ВЛГУ 09.03.01.ВТ-116.14.5.00

Словарь предметной области:

а) **Меню**

1) Атрибуты

- Изображения;

- Положение;

-Размеры

-Количество элементов параметров

2) Действия

-Отрисовка

-Считывание заданных данных

-Задание необходимых параметров объектам

-Начало симуляции

б) **Поле**

1) Атрибуты

- Изображения;

- Положение;

-Размеры

2) Действия

-Отрисовка

-Вывод общего количества волков и зайцев и их средн. возраст

-Задание расположения объектов на поле

в) **Волк**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

7

ВЛГУ 09.03.01.ВТ-116.14.5.00

1) Атрибуты

-Размер;

-Положение;

-Изображение;

-Возраст;

-Время размножения;

2) Действия

-Движение

-Размножение

-Проверить на столкновение;

-Смерть

г) **Заяц**

1) Атрибуты

-Размер;

-Положение;

-Изображение;

-Возраст;

-Время размножения;

2) Действия

-Движение

-Размножение

-Смерть

Результатом системного анализа является техническое задание на проектирование, составленное с использованием словаря предметной области.

Требования технического задания.

В моей задаче «Зайцы и волки», я моделирую процесс игры. В меню задаются все необходимые параметры для симуляции. Далее в поле появляются заданное число существ, которое начинает двигаться в хаотичном направлении, рекошетя от стен. В процессе игры зайцы сталкиваются с волками и уничтожаются. В определенный период времени зайцы и волки находящиеся в поле делятся, так же же ещё дают потомство. Через заданное время фигуры исчезают, так же ещё умирают от старости. Если в поле никого не остаётся (или только зайцы) симуляция прекращается.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

ВЛГУ 09.03.01.ВТ-116.14.5.00

При создании игры, я использовал: детальный текст задачи, язык программирования С++, операционную систему Windows, аппаратные средства IBM PC, среда для написания программы Microsoft Visual Studio.

3 Архитектура программной системы

3.1 Идентификация классов и объектов

Изм.

Лист

№ докум.

Подпись

Дата

Лист

9

ВЛГУ 09.03.01.ВТ-116.14.5.00

Для решения задачи используется четыре класса. Класс GodManager является главным управляющим классом который строится первоначальное поле и реализуется первичное меню для введения необходимых параметром, также класс совмещает поля с классом Object, который в свою очередь является классов родителем для классов Wolf и Rabbit. На основе полей дружественных классов

GodManager и Object через виртуальный метод AddNext классы наследники Wolf и Rabbit производят построение объектов (волков и зайцев).

3.2 Идентификация содержания классов

Класс GodManager : является управляющим классов «меню» содержащим введеную информацию о генерируемых объектах, так же друг класса Object

Методы класса:

GetID – получение ИД для элементов

SetBackGround – устанавливает фон приложения

GenerateObjects – генерация объектов в памяти

AddObject – добавление нового объекта

DeleteObjectofField – удаление объектов с поля

ActiveElements – активация элементов меню

DisabledElements – скрытие элементов меню

StartGame – начало симуляции игры

StopGame – завершение симуляции игры

GetInfoEdits – проверка введеных данных

AddElements – добавление элементов меню

PaintSettings – отрисовка значения настроек

PaintGame – отрисовка игры

Paint – отрисовка в зависимоти от режима меню или симуляция

Класс Object : родитель для классов Wolf и Rabbit содержит наследуемые методы, так же друг класса GodManager.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

10

ВЛГУ 09.03.01.ВТ-116.14.5.00

Методы класса:

Action – Проверка на пересечение

AddAction – Дополнительное действие

AddNext – Создание нового себе подобного объекта

GetID – Получение ИД объекта

Show – Отображение объектов через битовые изображения

Moving – Движение по полю объектов и проверка их на старост и рождение

Класс Wolf : содержит информацию об объектах волках

Методы класса:

Action – проверка на пересечение с зайцем при движении

AddNext – Добавление нового волка

AddAction – Отвечает за действие при пересечении зайца с волком

Класс Rabbit : задержит данные о зайцах

Методы класса:

Action – отсутствие дополнительных действий

AddNext – Добавление нового зайца

3.3 Реализация классов и объектов

Все разработанные классы и выявленные объекты заключены в проект. Уточненные диаграммы классов и объектов представлены в документах ВЛГУ 09.03.01.ВТ-116.14.5.01 и ВЛГУ 09.03.01.ВТ-116.14.5.02 соответственно. Алгоритм методов представлен в документе ВЛГУ 09.03.01.ВТ-116.14.5.03.

4 Инструкция

Игра написана и работает в среде Microsoft Visual Studio 2017. Для работы игры необходимо, либо запустить файл «Wolf\_Rabbit.exe», либо установить программу Microsoft Visual Studio 2017 и запустить файл «Wolf\_Rabbit.sln». После установки нужно нажать клавишу «Сборка», далее «Отладка».

Изм.

Лист

№ докум.

Подпись

Дата

Лист

11

ВЛГУ 09.03.01.ВТ-116.14.5.00

Появляется поле, на котором изображено стартовое меню для заполнения необходимыми значениями: количество зайцев и волков, продолжительность жизни зайцев и волков, время через которое появится новый заяц и волк, и период через который волк умрет от голода. При заполнении полей параметров и нажатии на кнопку начинается симуляция. На поле двигаются животные, а в верхнем правом углу указано количество и средний возраст зайцев и волков. Симуляция прекращается при исчезновении всех животных (или всех волков), и приложение возвращается в первоначальное меню.

5 Программирование

Программирование выполнено на языке С++, что соответствует общим требованиям к курсовой работе. Полный текст программы приведен ниже.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

12

ВЛГУ 09.03.01.ВТ-116.14.5.00

#include "Objects.h"

const char \* AppClassName = "MainFrame"; // класс окна

const UINT TIMER\_FRAME = 10; // Таймер

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam); // Обработчик сообщений

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrev, LPSTR cmd, int nCmd) // Главная функция

{

srand(time(0));

HWND hwnd; // Окно

MSG msg; // Сообщение

WNDCLASS wc; // Класс окна

wc.cbClsExtra = wc.cbWndExtra = 0; // Настройки класса окна

wc.hbrBackground = (HBRUSH)(BLACK\_BRUSH);

wc.hCursor = LoadCursor(NULL, IDC\_ARROW); // Курсор

wc.hIcon = LoadIcon(hInst, IDI\_APPLICATION);

wc.hInstance = hInst;

wc.lpfnWndProc = WndProc;

wc.lpszClassName = AppClassName;

wc.lpszMenuName = NULL;

wc.style = CS\_VREDRAW | CS\_HREDRAW;

RegisterClass(&wc); // Регистрация окна

hwnd = CreateWindow(AppClassName, "RoadCross", WS\_POPUP, 0, 0, 600, 600, NULL, NULL, hInst, NULL); // Создание окна

SetTimer(hwnd, TIMER\_FRAME, 15, NULL); // Установка таймера

GodManager::GetInstace().SetBackGround(IDB\_BACKGROUND); // Добавить фон

GodManager::GetInstace().AddElements(hwnd); // Добавить функц элементы

GodManager::GetInstace().ActiveElements(hwnd); // Активировать элементы

ShowWindow(hwnd, SW\_SHOW);

UpdateWindow(hwnd);

while (GetMessage(&msg, 0, 0, 0)) // Обработка сообщений

{

if (!IsDialogMessage(hwnd, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return msg.message;

}

LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)

{

HDC hdc;

PAINTSTRUCT ps;

switch (msg)

{

case WM\_COMMAND:

switch (wParam)

{

case 7: // Обработка кнопки

if (GodManager::GetInstace().GetInfoEdits(hwnd)) // Если удалось считать информацию с Editов

{

GodManager::GetInstace().StartGame(); // Начать игру

Изм.

Лист

№ докум.

Подпись

Дата

Лист

13

ВЛГУ 09.03.01.ВТ-116.14.5.00

GodManager::GetInstace().DisabledElements(hwnd); // Отключить элементы

}

break;

}

break;

case WM\_TIMER: // Сработал таймер

switch (wParam)

{

case TIMER\_FRAME:

InvalidateRect(hwnd, NULL, true); // Перерисовать

break;

}

break;

case WM\_ERASEBKGND:

return FALSE;

case WM\_PAINT: // Обработка отрисовки

hdc = BeginPaint(hwnd, &ps); // Начало отрисовки

GodManager::GetInstace().Paint(ps, 600, 600, hwnd);

EndPaint(hwnd, &ps); // Конец отрисовки

break;

case WM\_DESTROY: // Закрытие

PostQuitMessage(NULL);

return 0;

default:

return DefWindowProc(hwnd, msg, wParam, lParam);

}

return 0;

}

// далее Objects.h

#pragma once

#include <Windows.h>

#include <string>

#include <vector>

#include <chrono>

#pragma comment(lib,"MSIMG32.lib")

#pragma comment(lib, "Gdi32.lib")

#include "resource.h"

class Object;

class Wolf;

class Rabbit;

struct Settings // Настройки, получаемые из меню

{

int time\_eat\_wolf; // волк без еды

int time\_death\_wolf; // смерть волка

int time\_death\_rabbit; // смерть кролика

int time\_new\_wolf; // новый волк

int time\_new\_rabbit; // новый кролик

};

class GodManager // Главный класс управления

{

friend Object;

Settings settings; // Настройки

int index; // индекс для индентификации объекта

int count\_wolf; // количество волков

int count\_rabbit; // количество кроликов

bool game; // игра?

HBITMAP\* background; // фон

static GodManager \* manager; // сам элемент - Синглетон

std::vector<Object\*> objects; // объекты сцены

GodManager() // конструктор

Изм.

Лист

№ докум.

Подпись

Дата

Лист

14

ВЛГУ 09.03.01.ВТ-116.14.5.00

{

game = false;

index = 0;

}

public:

static UINT IDCsEdit[7]; // Идентификаторы Editов

static UINT Button\_ID; // Идентификатор кнопки

inline int GetID() // Получить ID для элемента

{

return index + 1;

}

inline void SetBackGround(WORD w\_hb) // Установить фон

{

background = new HBITMAP();

\*background = LoadBitmap(GetModuleHandle(NULL), MAKEINTRESOURCE(w\_hb));

}

void GenerateObjects(); // Сгенерировать в память все объекты сцены

inline void AddObject(Object\* obj) // Добавить объект

{

objects.push\_back(obj);

}

void DeleteObjectofField(Object\*); // Удалить объект

void ActiveElements(HWND hwnd) // Активировать элементы меню

{

for (int i = 0; i < 7; i++)

ShowWindow(GetDlgItem(hwnd, IDCsEdit[i]), SW\_SHOW);

ShowWindow(GetDlgItem(hwnd, Button\_ID), SW\_SHOW);

}

void DisabledElements(HWND hwnd) // Скрыть элементы

{

for (int i = 0; i < 7; i++)

ShowWindow(GetDlgItem(hwnd, IDCsEdit[i]), SW\_HIDE);

ShowWindow(GetDlgItem(hwnd, Button\_ID), SW\_HIDE);

}

inline void StartGame() // Старт

{

game = true;

this->GenerateObjects();

}

inline void StopGame() { game = false; } // Стоп игры

bool GetInfoEdits(HWND hwnd) // Получить инфо о введенных данных

{

char settings[7][10]; // для получения информации

for (int i = 0; i < 7; i++)

{

GetWindowText(GetDlgItem(hwnd, IDCsEdit[i]), settings[i], 10);

try // Если вдруг пустота или бред введен

{

std::stoi(std::string(settings[i]));

}

catch (std::invalid\_argument)

{

MessageBox(0, "Enter correct data", "Error", 0);

return false;

}

}

// Внесение данных

this->count\_wolf = std::stoi(std::string(settings[0]));

this->count\_rabbit = std::stoi(std::string(settings[1]));

this->settings.time\_death\_wolf = std::stoi(std::string(settings[2]));

this->settings.time\_death\_rabbit = std::stoi(std::string(settings[3]));

this->settings.time\_new\_wolf = std::stoi(std::string(settings[4]));

this->settings.time\_new\_rabbit = std::stoi(std::string(settings[5]));

this->settings.time\_eat\_wolf = std::stoi(std::string(settings[6]));

return true;

}

void AddElements(HWND hwnd) // Добавить элементы

Изм.

Лист

№ докум.

Подпись

Дата

Лист

15

ВЛГУ 09.03.01.ВТ-116.14.5.00

{

int i;

for (i = 0; i < 7; i++)

{

HWND hwnd\_t = CreateWindowEx(0, "EDIT", "", WS\_CHILD | WS\_BORDER | ES\_LEFT | ES\_AUTOHSCROLL,

300, 100 + i \* 50, 50, 20, hwnd, (HMENU)IDCsEdit[i], GetModuleHandle(NULL), NULL);

}

HWND hwnd\_t = CreateWindow("BUTTON", "Start", WS\_CHILD, 300, 100 + i \* 50, 70, 20, hwnd, (HMENU)Button\_ID, GetModuleHandle(NULL), NULL);

}

void PaintSettings(HDC hdc) // Отрисовать настройки

{

std::string strs[7] = { "Count Wolf","Count Rabbit","Wolf's lifetime","Rabbit's lifetime","Wolf's new","Rabbit's new","Wolf's hungry", };

for (int i = 0; i < 7; i++)

TextOut(hdc, 150, 100 + i \* 50, strs[i].c\_str(), strs[i].length());

}

void PaintGame(HDC hdc, HWND hwnd); // Отрисовать игру

void Paint(PAINTSTRUCT& ps, int width, int height, HWND hwnd) // Отрисовать

{

HDC hMemDC, hTempDC; // Теневой буффер

HGDIOBJ hMemBmp, hSysBmp;

HBRUSH hBrush = (HBRUSH)WHITE\_BRUSH;

hMemDC = CreateCompatibleDC(ps.hdc); // Создать конекст схожии с основным

hMemBmp = CreateCompatibleBitmap(ps.hdc, width, height); // Создание битмап для фона

hSysBmp = SelectObject(hMemDC, hMemBmp);

SelectObject(hMemDC, hBrush);

hTempDC = CreateCompatibleDC(hMemDC);

SelectObject(hTempDC, \*background);

BitBlt(hMemDC, 0, 0, width, height, hTempDC, 0, 0, SRCCOPY); // Копироание одного контекста в другой контекст

DeleteDC(hTempDC); // Удаление временного контекста

if (game) // Если игра

this->PaintGame(hMemDC, hwnd);

else

this->PaintSettings(hMemDC);

BitBlt(ps.hdc, 0, 0, width, height, hMemDC, 0, 0, SRCCOPY); // Отправка в основной контекст

SelectObject(hMemDC, hSysBmp);

DeleteObject(hMemBmp); // Удаление Битмапа

DeleteObject(hSysBmp);

DeleteDC(hMemDC); // Удаление контекстов

DeleteObject(hBrush);

}

static GodManager & GetInstace() // Получить ссылку на объект

{

if (!manager)

manager = new GodManager();

return \*manager;

}

};

class Object // Объект

{

friend GodManager;

private:

float Lenght(float x1, float y1, float x2, float y2) // Длина вектора

{

return sqrt((x2 - x1)\*(x2 - x1) + (y2 - y1)\*(y2 - y1));

}

protected:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

16

ВЛГУ 09.03.01.ВТ-116.14.5.00

int id; // ИД

static HBITMAP hbitmaps[2]; // битмапы

int type\_hb; // тип используемого битмапа

std::pair<float, float> position; // позиция

std::pair<float, float> velocity; // ускорение

BITMAP info; // информация о битмапеы

std::chrono::steady\_clock::time\_point last\_nlife, startlife; // последнее рождение и начало жизни

int age; // возраст

int time\_death; // смертельный возраст

int time\_new; // время рождения

virtual void Action(Object\*) = 0; // Действие

Object\* Contact(std::vector<Object\*>& objects) // Проверка на пересечение

{

for (UINT i = 0; i < objects.size(); i++)

if (objects[i]->id != id)

{

int len = Lenght(position.first + info.bmWidth / 2, position.second + info.bmHeight / 2,

objects[i]->position.first + objects[i]->info.bmWidth / 2,

objects[i]->position.second + objects[i]->info.bmHeight / 2);

len -= (this->info.bmWidth / 2);

if (len < 0)

return objects[i];

}

return NULL;

}

virtual void AddAction() {} // Дополнительно действие

virtual Object\* AddNext() = 0; // Создать нового себя

public:

virtual ~Object() // Деструктор

{

GodManager::GetInstace().DeleteObjectofField(this);

}

Object(int type, int index, std::pair<float, float> pos) :type\_hb(type), id(index) // Констурктор

{

GetObject(hbitmaps[type\_hb], sizeof(BITMAP), &info); // Получить информацию о битмапе

startlife = last\_nlife = std::chrono::high\_resolution\_clock::now();

if (pos.first == -1)

{

position.first = 50 + rand() % 500;

position.second = 50 + rand() % 500;

}

else

{

position.first = pos.first;

position.second = pos.second;

}

velocity.first = -3 + rand() % 7;

velocity.second = -3 + rand() % 7;

float size = sqrt(velocity.first\*velocity.first + velocity.second\*velocity.second);

velocity.first /= size;

velocity.second /= size;

}

inline int GetID() { return id; } // Получить ИД

void Show(HDC hdc) // Отобразить

{

HDC hMem = CreateCompatibleDC(hdc); // Создать схожий контекст

SelectObject(hMem, hbitmaps[type\_hb]); // Выбрать в него изображение

// Скпировать с исключением черного цвета

TransparentBlt(hdc, position.first, position.second, info.bmWidth, info.bmHeight, hMem, 0, 0,

info.bmWidth, info.bmHeight, RGB(0, 0, 0));

DeleteDC(hMem); // Удалить ненужный контекст

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

17

ВЛГУ 09.03.01.ВТ-116.14.5.00

void Moving(std::vector<Object\*>& objects) // движение

{

// Проверка на старость

if (std::chrono::duration\_cast<std::chrono::seconds>(std::chrono::high\_resolution\_clock::now() - startlife).count() > time\_death - age)

{

this->~Object();

return;

}

// Проверка на рождение

if (std::chrono::duration\_cast<std::chrono::seconds>(std::chrono::high\_resolution\_clock::now() - last\_nlife).count() > time\_new)

{

GodManager::GetInstace().AddObject(this->AddNext());

last\_nlife = std::chrono::high\_resolution\_clock::now();

}

// Удар о стены

if (position.first >= 600 - info.bmWidth || position.first <= 10)

velocity.first \*= -1;

if (position.second >= 600 - info.bmHeight || position.second <= 10)

velocity.second \*= -1;

position.first += velocity.first \* 5;

position.second += velocity.second \* 5;

// Проверка на контакт

Object \* temp = Contact(objects);

if (temp != NULL)

Action(temp);

AddAction();

}

};

class Wolf :public Object // Волк

{

std::chrono::steady\_clock::time\_point last\_eat; // время есть

int time\_eat; // максимальный голод

public:

// Конструктор

Wolf(int index, int \_time\_eat, int \_time\_death, int \_time\_new, std::pair<float, float> pos = { -1,-1 }) :Object(0, index, pos)

{

last\_eat = std::chrono::high\_resolution\_clock::now();

this->time\_eat = \_time\_eat;

this->time\_death = \_time\_death;

this->time\_new = \_time\_new;

age = 1 + rand() % time\_death;

}

// Действие

void Action(Object\* obj);

Object\* AddNext() // Создать себеподобного

{

return new Wolf(GodManager::GetInstace().GetID(), time\_eat, time\_death, time\_new, position);

}

void AddAction() // Доп действие - проверка на голод

{

if (std::chrono::duration\_cast<std::chrono::seconds>(std::chrono::high\_resolution\_clock::now() - last\_eat).count() > time\_eat)

this->~Wolf();

}

};

class Rabbit :public Object // Кролик

{

public:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

18

ВЛГУ 09.03.01.ВТ-116.14.5.00

// Конструктор

Rabbit(int index, int \_time\_death, int \_time\_new, std::pair<float, float> pos = { -1,-1 }) :Object(1, index, pos)

{

this->time\_death = \_time\_death;

this->time\_new = \_time\_new;

age = 1 + rand() % time\_death;

}

void Action(Object\* obj) {} // Нет никаких действий

Object\* AddNext() // Создать нового потомка

{

return new Rabbit(GodManager::GetInstace().GetID(), time\_death, time\_new, position);

}

};

// Objects.cpp

#include "Objects.h"

GodManager \* GodManager::manager = nullptr; // определение статики

HBITMAP Object::hbitmaps[2]{ LoadBitmap(GetModuleHandle(NULL),MAKEINTRESOURCE(IDB\_WOLF)),

LoadBitmap(GetModuleHandle(NULL),MAKEINTRESOURCE(IDB\_RABBIT)) }; // статика битмапов

UINT GodManager::IDCsEdit[7]{ 0,1,2,3,4,5,6 }; // статика для ИД Editов

UINT GodManager::Button\_ID = 7; // статика ИД кнопки

void GodManager::DeleteObjectofField(Object\* obj) // Удалить объект со сцены

{

int i;

for (i = 0; i < objects.size(); i++)

if (obj->GetID() == objects[i]->GetID())

break;

if (i == objects.size())

return;

objects.erase(objects.begin() + i);

}

void GodManager::GenerateObjects() // Сгенерировать объекты в память по количеству

{

if (objects.size() != 0)

{

for (int i = 0; i < objects.size(); i++)

{

objects[i]->~Object();

objects.erase(objects.begin() + i);

i--;

}

}

index = 0;

for (int i = 0; i < count\_wolf; i++)

objects.push\_back(new Wolf(index++, settings.time\_eat\_wolf, settings.time\_death\_wolf, settings.time\_new\_wolf));

for (int i = 0; i < count\_rabbit; i++)

objects.push\_back(new Rabbit(index++, settings.time\_death\_rabbit, settings.time\_new\_rabbit));

}

void GodManager::PaintGame(HDC hdc, HWND hwnd) // Отрисовать игру

{

int count\_r = 0, count\_w = 0, a\_r = 0, a\_w = 0;

for (int i = 0; i < objects.size(); i++) // Подсчет количества для вывод статистики

{

if (typeid(Rabbit) == typeid(\*objects[i]))

Изм.

Лист

№ докум.

Подпись

Дата

Лист

19

ВЛГУ 09.03.01.ВТ-116.14.5.00

{

count\_r++;

a\_r += objects[i]->age;

}

else

{

count\_w++;

a\_w += objects[i]->age;

}

objects[i]->Show(hdc);

objects[i]->Moving(objects);

}

std::string info\_r = "", info\_w = "";

if (count\_r != 0)

info\_r = "Rabbit count/average age - " + std::to\_string(count\_r) + "/" + std::to\_string((float)a\_r / (float)count\_r);

if (count\_w != 0)

info\_w = "Wolf count/average age - " + std::to\_string(count\_w) + "/" + std::to\_string((float)a\_w / (float)count\_w);

TextOut(hdc, 350, 20, info\_r.c\_str(), info\_r.length());

TextOut(hdc, 350, 40, info\_w.c\_str(), info\_w.length());

if (count\_r + count\_w == 0) // Если количество элементов =0

{

this->StopGame();

this->ActiveElements(hwnd);

}

}

void Wolf::Action(Object\* obj) // Действия волков по отношению к зайцам

{

if (typeid(Rabbit) == typeid(\*obj))

{

GodManager::GetInstace().DeleteObjectofField(obj);

this->last\_eat = std::chrono::high\_resolution\_clock::now();

}

}

Изм.

Лист

№ докум.

Подпись

Дата

Лист

20

ВЛГУ 09.03.01.ВТ-116.14.5.00

6 Тестирование

На рисунках 1, 2, 3, 4 представлено тестирование программы.

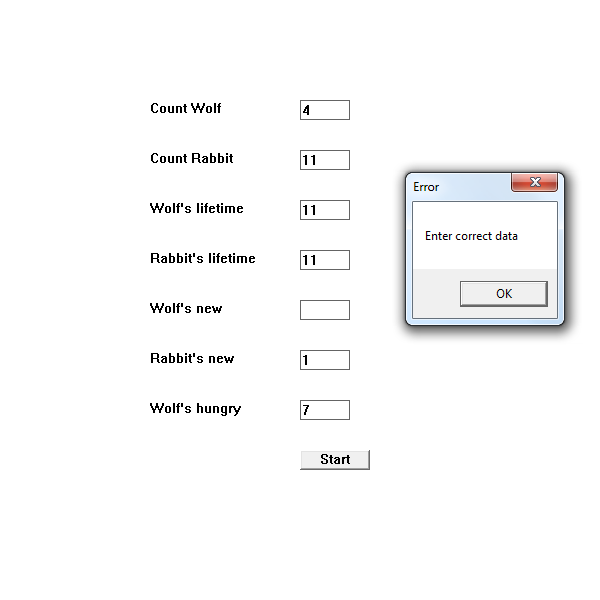
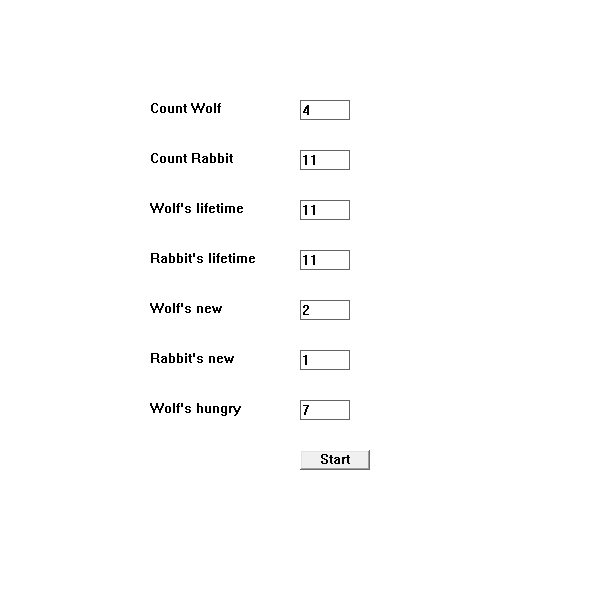


Рисунок 1 – меню Рисунок 2 – ошибка при пустом поле



Рисунок 3 – начало модуляции Рисунок 4 – через некоторое время

Заключение

Изм.

Лист

№ докум.

Подпись

Дата

Лист

21

ВЛГУ 09.03.01.ВТ-116.14.5.00

В результате выполнения работы, была получена игра «Зайцы и волки». Данная игры была написана на языке С++, в объектно-ориентированной среде Microsoft Visual Studio. В операционной системе Windows, аппаратные средства IBM PC. Программа полностью соответствует техническому заданию. Для программы написаны алгоритмы методов, диаграмма классов и объектов. Программа содержит 477 строк. Время выполнения программы от 1 минуты. Размер файла .exe составляет 1,4 Мб. Размер файла .sln 2 КБ. Размер файла .cpp

6 КБ.

Список использованных источников

Изм.

Лист

№ докум.

Подпись

Дата

Лист

22

ВЛГУ 09.03.01.ВТ-116.14.5.00

1. И. Р. Дубов, В. А. Барков, А. Г. Долин. Технология программирования. В.:Владим. Гос. Ун-т., 1999. – 48 с.
2. И. Р. Дубов, В. В. Власенко. Объектно-ориентированное программирование. В.: Владим. Гос. Ун-т., 2003. – 65 с.
3. В. В. Подбельский. Язык СИ++. М.: Финансы и статистика. 2004. -560 с.
4. Р. Лафоре. Объектно-ориентированное программирование в С++. СПб:Питер. 2004.- 928 с.
5. А. Ю. Щупак. Win32 API. Разработка приложений для Windows. СПб:Питер. 2008.- 592 с.

|  |
| --- |
|  |

Изм.

Лист

№ Докум.

Подпись

Дата

Масса

Лист 1

ВЛГУ 09.03.01.ВТ-116.14.5.01

Разработал

Лаптев Д.С.

Проверил

Быков В. И.

Н. Контр.

Утвердил

Курсовая работа.

Диаграмма классов

Лит.

Масштаб

Листов 1

ВТ-116

У

Т. Контр.

Реценз.

|  |
| --- |
| Object |
| Id  Hbitmaps  Position  Velocite  Info  Last\_nkife  Startlife  Age  Time\_death  Time\_new  Lenght |
| Action  AddAction  AddNext  GetID  Moving |

|  |
| --- |
| GodManager |
| Setting  Index  Count\_wolf  Count\_rabbit  Game  Background |
| GetID  SetBackGround  GenerateObjects  AddObject  DeleteObjectofField ActiveElements  DisabledElements  StartGame  StopGame  GetInfoEdits  AddElements  PaintSettings  PaintGame  Paint |

|  |
| --- |
| Wolf |
| Time\_death  Time\_eat  Time\_new  Age |
| Action  AddNext  AddAction |

|  |
| --- |
| Rabbit |
| Time\_death  Time\_new  Age |
| Action  AddNext |

|  |
| --- |
|  |

Изм.

Лист

№ Докум.

Подпись

Дата

Масса

Лист 1

ВЛГУ 09.03.01.ВТ-116.14.5.02

Разработал

Лаптев Д.С.

Проверил

Быков В. И.

Н. Контр.

Утвердил

Курсовая работа.

Диаграмма объектов

Лит.

Масштаб

Листов 1

ВТ-116

У

Т. Контр.

Реценз.

|  |
| --- |
| Object |
| Id  Hbitmaps  Position  Velocite  Info  Last\_nkife  Startlife  Age  Time\_death  Time\_new  Lenght |
| Action  AddAction  AddNext  GetID  Moving |

|  |
| --- |
| GodManager |
| Setting  Index  Count\_wolf  Count\_rabbit  Game  Background |
| GetID  SetBackGround  GenerateObjects  AddObject  DeleteObjectofField ActiveElements  DisabledElements  StartGame  StopGame  GetInfoEdits  AddElements  15  PaintSettings  PaintGame  Paint |

11

4

|  |
| --- |
| Wolf |
| Time\_death  Time\_eat  Time\_new  Age |
| Action  AddNext  AddAction |

|  |
| --- |
| Rabbit |
| Time\_death  Time\_new  Age |
| Action  AddNext |

Изм.

Лист

№ Докум.

Подпись

Дата

Масса

Лист 1

ВЛГУ 09.03.01.ВТ-116.14.5.03

Разработал

Лаптев Д.С.

Проверил

Быков В. И.

Н. Контр.

Утвердил

Курсовая работа.

Алгоритм методов.

Лит.

Масштаб

Листов 1

ВТ-116

У

Т. Контр.

Реценз.

Начало

Создание поля,

введение параметров

Генерация расположения волков и зайцев

Отрисовка

Конец

Выполнение процесса моделирования

Выход?

Да

Нет