МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ЧУО «Минский КОЛЛЕДЖ предпринимательства»

**морской бой**

Пояснительная записка к курсовому проекту по учебной дисциплине

«Основы алгоритмизации и программирование»

КПП22.018.104.081ПЗ

**Автор проекта**

Учащийся 2 курса группы П-22 Д. О. Кириченко

**Руководитель**

Преподаватель О. А. Салтыкова

Минск, 2018МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ЧУО «МИНСКИЙ КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА»

Специальность

«Программное обеспечение информационных технологий» 2-40 01 01

УТВЕРЖДАЮ

Председатель цикловой комиссии

\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

**ЗАДАНИЕ**

**на курсовой проект**

Учащемуся Кириченку Данилу Олеговичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

курса 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ группы П22\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

по дисциплине «Основы алгоритмизации и программирование» \_\_\_\_\_\_\_\_\_\_\_

Тема курсового проекта Морской бой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исходные данные OOP\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Состав проекта**

**Пояснительная записка**

Содержание разделов Срок выполнения

Введение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1 Постановка задачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2 Вычислительная система\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3 Проектирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4 Описание программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Отладка и испытание программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6 Описание применения программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Заключение\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Графическая часть проекта**

Схема работы программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи «01» апреля\_\_\_\_\_\_2018г.

Срок завершения «06» июня\_\_\_2018г.

**Преподаватель-руководитель курсового проекта** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись учащегося**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **СОДЕРЖАНИЕ**  [Введение 4.](#_Toc422206439)  [1 Постановки задачи 6.](#_Toc422206440)  [1.1 Формулировка задания 6.](#_Toc422206441)  [1.2 Определение данных и их представление 6.](#_Toc422206442)  [2 Вычислительная система 7.](#_Toc422206443)  [2.1 Обоснование выбора языка программирования 7.](#_Toc422206444)  [2.2 Обоснование выбора среды разработки](#_Toc422206445) 7[. Требования к конфигурации программного и аппаратного обеспечения Ошибка! Закладка не определена.](#_Toc422206446)  [2.3 Требования к конфигурации программного и аппаратного обеспечения](#_Toc422206445) [8. Требования к конфигурации программного и аппаратного обеспечения Ошибка! Закладка не определена.](#_Toc422206446)  [3 Проектирование 9.](#_Toc422206447)  [3.1 Проектирование интерфейса 9.](#_Toc422206448)  [3.2 Физическая модель базы данных 11.](#_Toc422206449)  [4 Описание программы 12.](#_Toc422206451)  [4.1 Логическая структура 12.](#_Toc422206452)  [4.2 Физическая структура 18.](#_Toc422206453)  [4.3 Особенности интерфейса 20.](#_Toc422206454)  [5 Отладка и испытание программы 24.](#_Toc422206455)  [5.1 Тестовые примеры 24.](#_Toc422206456)  [5.2 Анализ полученных результатов 27.](#_Toc422206457)  [6 Описание применения программы 29.](#_Toc422206458)  [6.1 Назначение программы и условия применения 29.](#_Toc422206459)  [Заключение 30.](#_Toc422206460)  [Список использованных источников 32.](#_Toc422206461)  [Приложение 1 33.](#_Toc422206462) | | | | | | | | | | |  |
|  |  |  |  |  | *КПП22.018.104.081ПЗ* | | | | | | |
|  |  |  |  |  |
| *Изм* | *Лист* | *№ докум.* | *Подпись* | *Дата* |
| *Разраб.* | | *Яблонский* |  |  | *Алгебра матриц* | *Лит.* | | | *Лист* | *Листов* | |
| *Пров.* | | *Салтыкова* |  |  |  | *у* |  | 3 | 68 | |
| *Реценз.* | |  |  |  | *ЧУО МКП* | | | | | |
| *Н.Контр.* | |  |  |  |
| *Утв.* | |  |  |  |

ВВЕДЕНИЕ

В данном курсовом проекте необходимо разработать программное средство «морской бой». Курсовой проект предназначен для развлекательных целий.

Игра для двух участников, в которой игроки по очереди стреляют по координатам поля. Если у соперника по этим координатам имеется корабль (координаты заняты), то корабль или его часть «топится». Цель игрока — первым потопить все корабли противника.

Основной целью выполнения курсового проекта является расширение, углубление знаний по дисциплине «Основы алгоритмизации и программирование», формирование навыков научно-исследовательской деятельности, а также с базовыми принципами OOP.

Задачи выполнения курсового проекта:

* систематизация знаний в области программирования на языке Pascal.
* углубление уровня и расширение объема профессионально значимых знаний, умений и навыков разработки программ в среде Delphi;
* формирование умений и навыков самостоятельной организации научно-исследовательской работы;

- овладение современными методами поиска, обработки и использования информации.

Данная пояснительная записка содержит следующие разделы:

1. Раздел «Постановка задачи» содержит описание всех функций проекта, входных и выходных данных.
2. В разделе «Вычислительная система» приводится описание операционной системы, языка программирования и среды разработки выбранных для реализации проекта.
3. Раздел «Проектирование» содержит описание внешнего пользовательского интерфейса, описание алгоритма программы и структуры справочной системы.
4. В разделе «Описание программы» описываются все библиотечные модули, входящие в состав программы, а также назначение всех функций, входящих в приложение, здесь же приводится интерфейс программы с рисунками пользовательских форм.
5. Раздел «Отладка и испытание программы» иллюстрирует проверку работы программы на различных тестах и содержит анализ полученных результатов.
6. В разделе «Описание применения программы» содержатся сведения о назначении программного средства, классе решаемых задач и ограничениях, накладываемых на область применения.
7. В «Заключении» описываются использованные методы и средства, полнота раскрытия проблемы и возможность дальнейшей модификации проекта.
8. ПОСТАНОВКА ЗАДАЧИ
   1. Формулировка задания

Необходимо разработать программное средство «Морской Бой». Реализовать следующие операции:

1. Динамическое создание поля
2. Динамическое создание кораблей
3. Генерация Кораблей
4. Слабый искусственный интеллект
5. Использование OOP
6. Реализовать многопоточную программу

Предусмотреть возможность заполнения поля автоматически (генерируя случайным образом).

Разработать справочную систему для приложения.

* 1. Определение данных и их представление

Входные данные:

* Размерность поля, Количество кораблей;

Выходные данные:

* Визуализация взаимодействия бота и пользователя

1. Вычислительная система

**2**

* 1. Обоснование выбора языка программирования

Pascal – один из наиболее известных языков программирования, который очень эффективен для обучения структурному программированию, благодаря небольшому количеству синтаксических конструкций.

Особенностями языка является строгая типизация и наличие средств структурного программирования. Pascal способствует дисциплинированному программированию, так как на ряду со строгой типизацией там сведены к минимуму возможные синтаксические неоднозначности, при этом синтаксис языка интуитивно понятен.

Диалект Object Pascal разработанный в 1985 году поддерживает объектно-ориентированное программирование.

* 1. Обоснование выбора среды разработки

Delphi – интегрированная среда разработки программного обеспечения для операционной системы Windows.

Высоко производительный инструмент визуального построения приложений включает в себя компилятор кода и предоставляет средства визуального программирования. В Delphi так же входят локальный SQL-сервер, генераторы отчетов, библиотеки компонентов.

Преимущества Delphi:

* Быстрая разработка приложений
* Высокая производительность разработанных приложений
* Низкие требование разработанного приложения к ресурсам компьютера
* Наращиваемость за счет встраивания новых компонент в среду Delphi
* Возможность разработки новых компонентов собственными средствами Delphi
* Удачная проработки иерархии объектов
  1. Требования к конфигурации программного и аппаратного обеспечения

Windows 7 – пользовательская операционная система семейства Windows NT. В этой операционной системе реализована поддержка Unicode 5.1, она обладает поддержкой мультитач-управления.

Дополнительным преимуществом Windows 7 является более тесная интеграция с производителями драйверов, 50 новых шрифтов, улучшенная совместимость со старыми приложениями.

Сетевая технология Branch Cache позволяет кэшировать содержимое интернет трафика.

Минимальные требования:

- процессор Intel Pentium III 420 Мгц

- ОЗУ 256 Мбайт

- Место на HDD 1024 Мбайт

1. Проектирование
   1. Проектирование интерфейса

Главная форма приложения содержит 3 компонентов типа TButton:

* Начать Игру
* Настройки
* Выход

При нажатии на кнопку «Начать Игру», открывается новая форма «Gams», на которой расположена 1 кнопка (Button) и 2 панели (Panel1) на которых динамически создаться Image в зависимости от настроек на форме Option.

При нажатии на кнопку «Настройки» открывается новая форма «Option», на которой расположен RadioGroup с основными настройками и кнопка Button для сохранения настроек.

При нажатии на кнопку «Выход» можно выйти из программы

* 1. Описание алгоритма

Игровое поле — n×n у каждого игрока, на котором размещается флот кораблей. Размещаются:

* n корабль — ряд из 4 клеток («четырёхпалубный»; линкор)
* n корабля — ряд из 3 клеток («трёхпалубные»; крейсера)
* n корабля — ряд из 2 клеток («двухпалубные»; эсминцы)
* n корабля — 1 клетка («однопалубные»; торпедные катера)

Другой вариант названия: -трубные (напр. двухтрубные).

При размещении корабли не могут касаться друг друга сторонами и углами.

Рядом со «своим» полем чертится «чужое» такого же размера, только пустое. Это участок моря, где плавают корабли противника.

При попадании в корабль противника — на чужом поле ставится крестик, при холостом выстреле — закрашивается квадрат.

Самыми уязвимыми являются линкор и торпедный катер: первый из-за крупных размеров, в связи с чем его сравнительно легко найти, а второй из-за того, что топится с одного удара, хотя его найти достаточно сложно.

1. Описание программы
   1. Логическая структура

Основная форма (Form2):

procedure FormCreate(Sender: TObject); - создание формы

procedure Button1Click(Sender: TObject); - Начать Игру, открыть Form1

procedure Button2Click(Sender: TObject); - Настройки, Открыть Form3

procedure Button3Click(Sender: TObject); - Выход, Выйти из програмы

Форма(Form3):

procedure FormCreate(Sender: TObject); - создание формы

procedure СохранитьClick(Sender: TObject); - Сохранить, перейти на Form2

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean); - При нажатии на крестик перейти на Form2.

Форма(Form1):

procedure FormCreate(Sender: TObject); - создание формы, генерация поля в зависимости от настроек Form3, а также генерация в память кораблей противника.

procedure BitBtn1Click(Sender: TObject); - Генерация кораблей пользователя и начало игры.

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean); - при нажатие на крестик выход из программы.

procedure ImageOnClick(Sender: TObject); - событие для всех кликов пользователя.

procedure BotImageOnClick(Sender: TObject); - событие для всех кликов

бота.

* 1. Физическая структура

Unit GnShipUser; - – генерация кораблей игрока и поля в потоке.

Unit GnShip – генерация кораблей противника в потоке.

Unit GnTopImageBOT - генерация поля противника в потоке.

Unit MyTypes - вспомогательные массивы данных.

Unit PShip1x1 – содержит класс кораблей 1 клетка («однопалубные»; торпедные катера)

Unit PShip2x2 - содержит класс кораблей из 2 клеток («двухпалубные»; эсминцы)

Unit Pship3x3 - содержит класс кораблей из 3 клеток («трёхпалубные»; крейсера)

Unit Pship4x4; - содержит класс кораблей из 4 клеток («четырёхпалубный»; линкор)

Unit PImage; - содержит класс наследованный от TImage с дополнительными координатами Fx, Fy.

* 1. Особенности интерфейса

Интерфейс главной формы приложения показан на рисунке 1

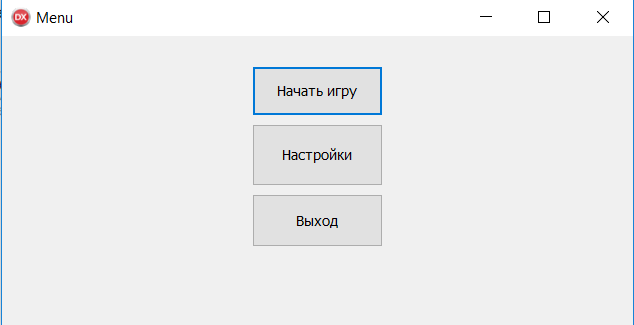


Рисунок 1

Интерфейс формы с игрой показан на рисунках 2.1, 2.2, 2.3

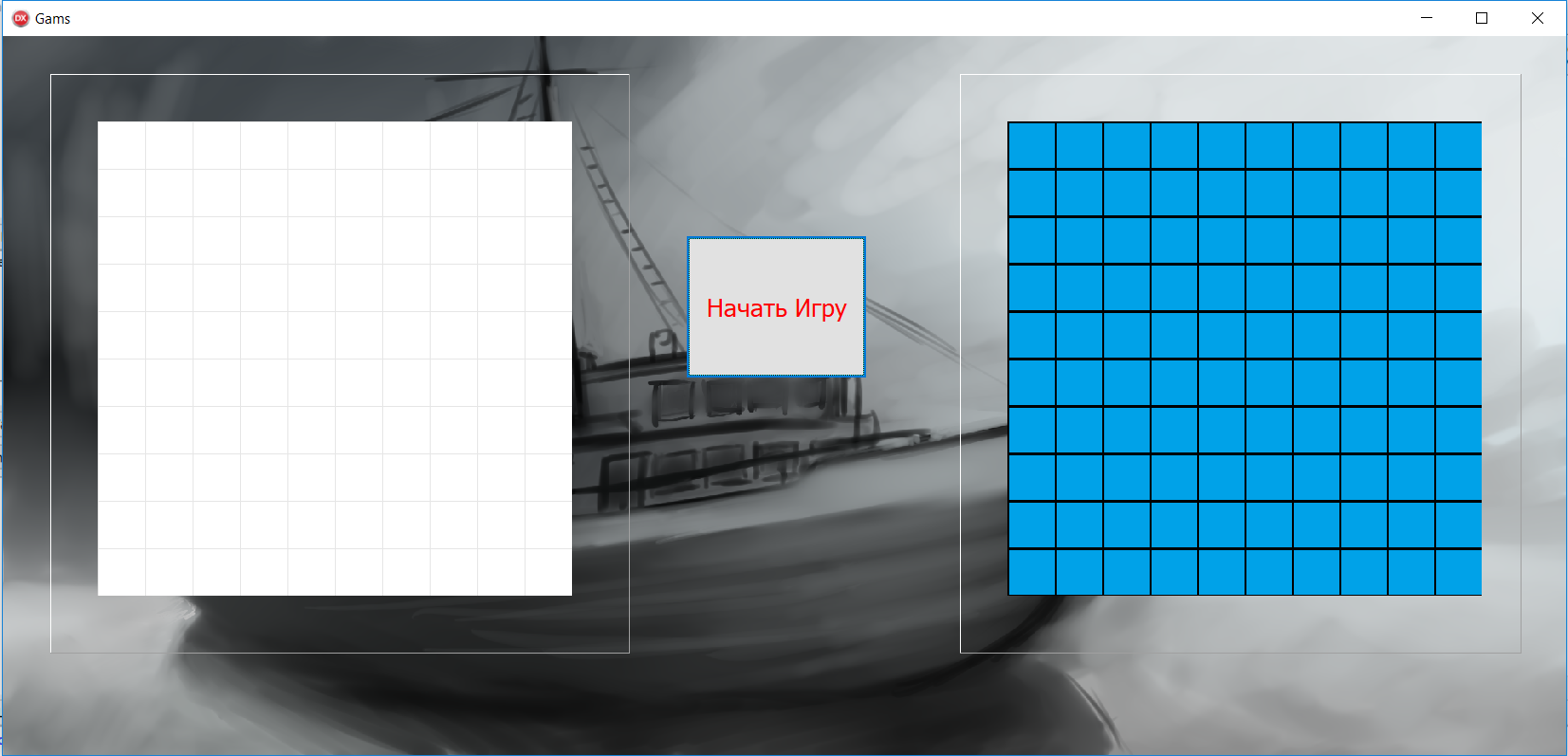


Рисунок 2.1

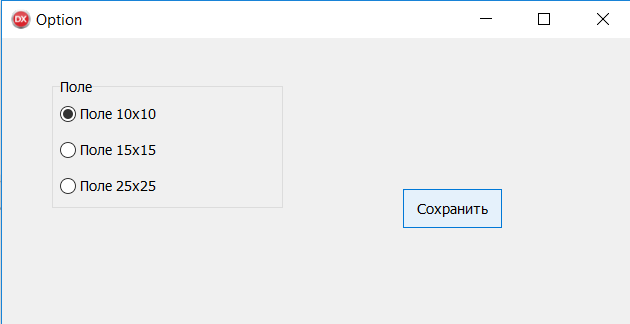


Рисунок 2.2



Рисунок 2.3

Интерфейс формы с настройками показан на рисунке 3



Рисунок

1. ОТЛАДКА И ИСПЫТАНИЕ ПРОГРАММЫ
   1. Тестовые примеры

Тест №1: Создание полей. Рисунки 4.1, 4.2

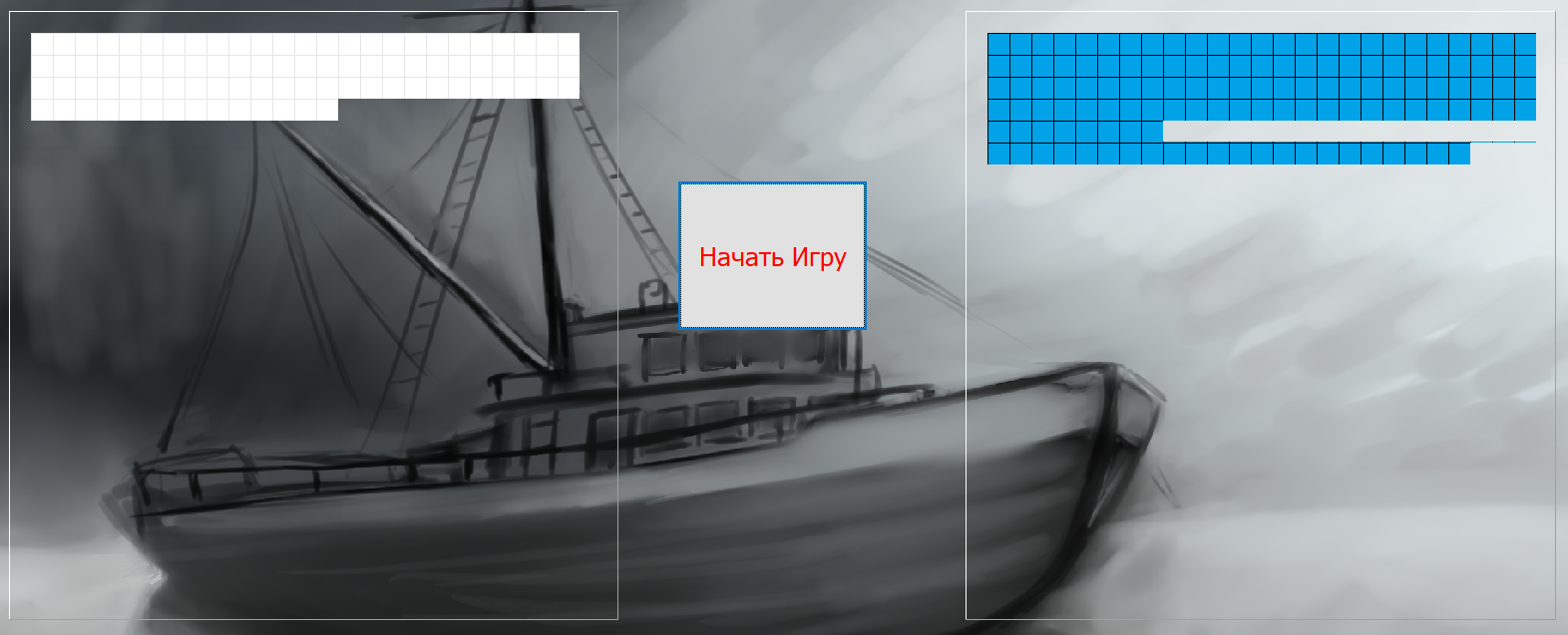


Рисунок 4.2 – Тест №1

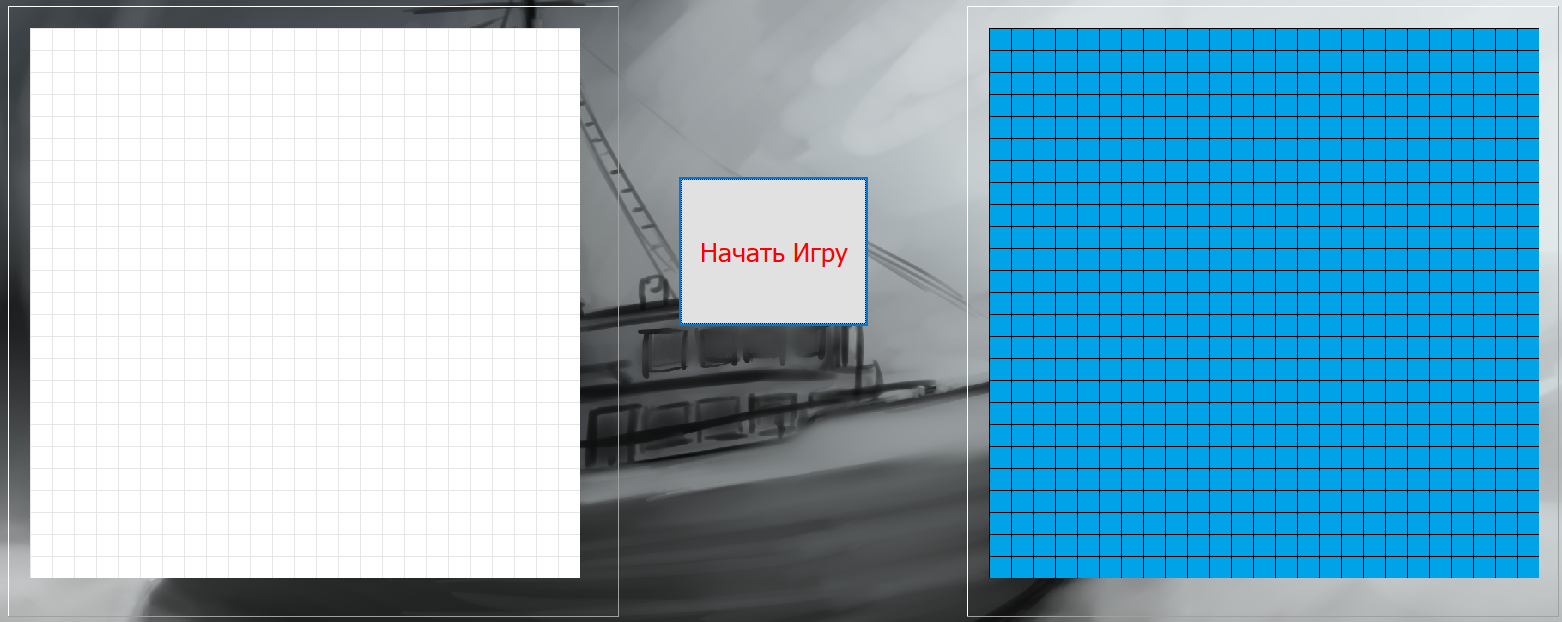
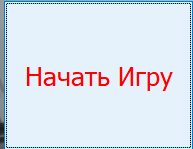


Рисунок 4.2 – Тест №1

Тест №2: Генерация Кораблей Пользователя. Рисунки 5.1, 5.2, 5.3



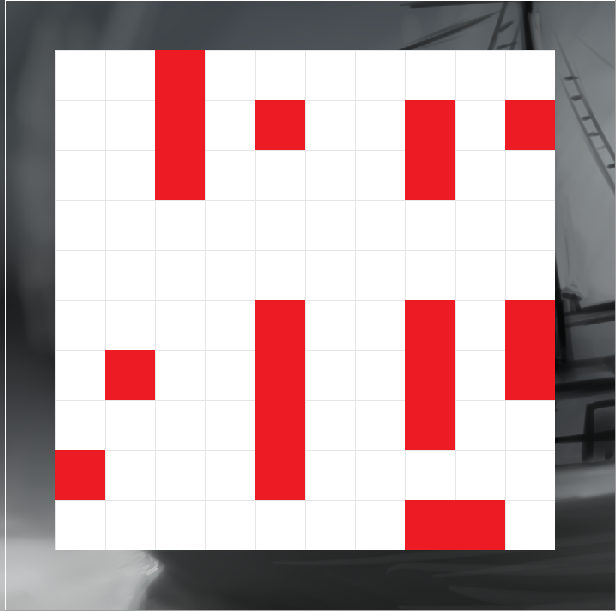


Рисунок 5.1 – Тест №2

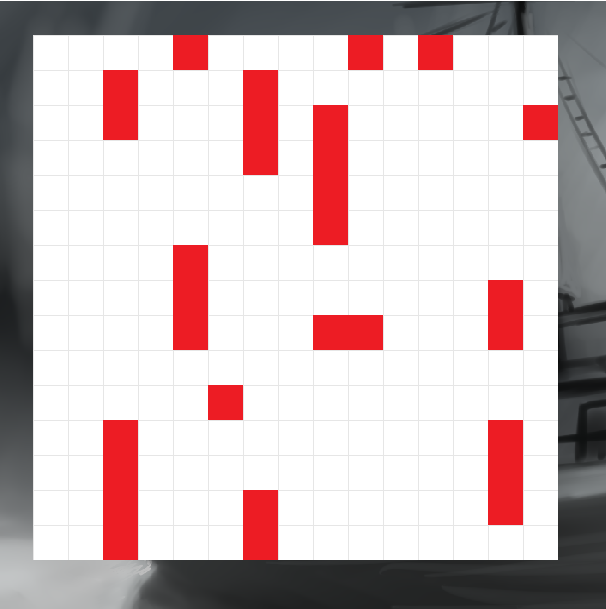


Рисунок 5.2 – Тест №2

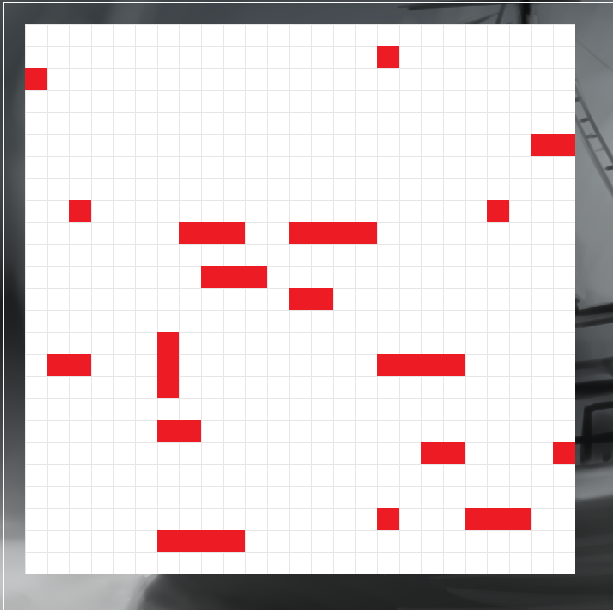


Рисунок 5.3 – Тест №2

Тест №3: Клик по полю. Рисунок 6.

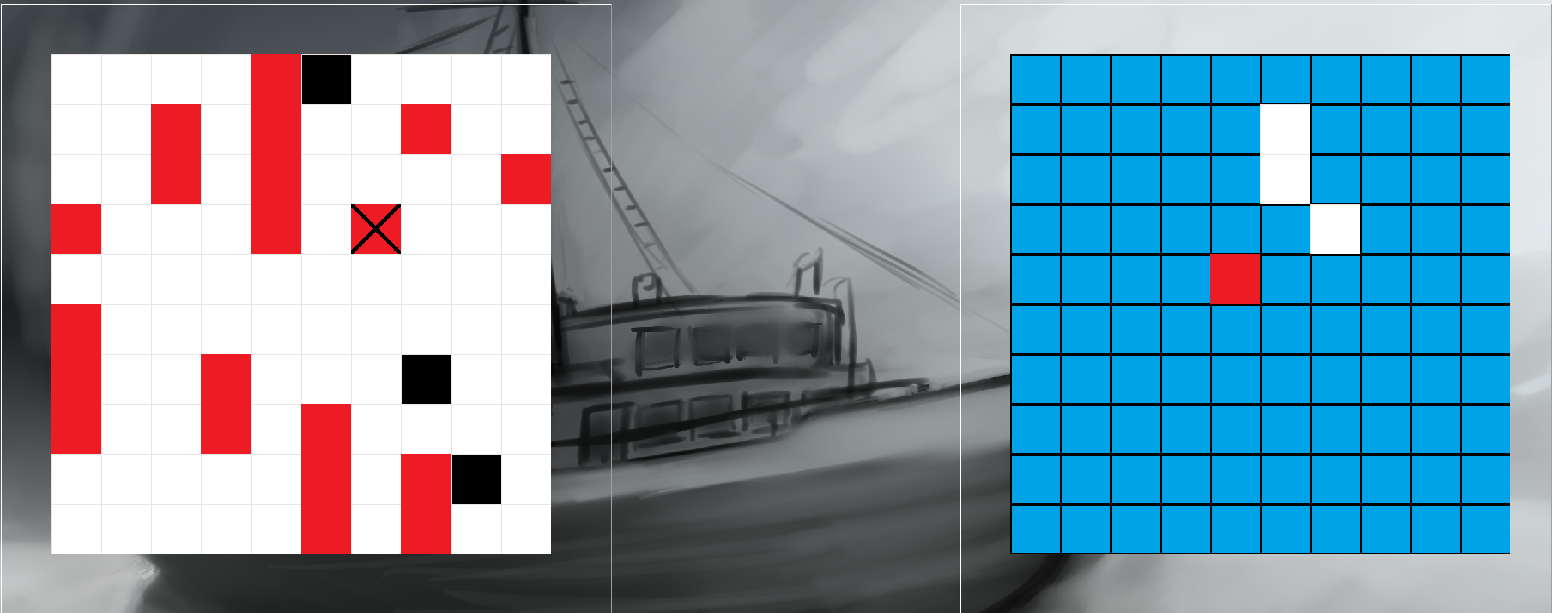


Рисунок 6 – Тест №3

* 1. Анализ полученных результатов

Тест №1 показал, что поле генерируется динамически и без ошибок.

Тест №2 показал, что корабли пользователя генерируются динамически и на разных полях без ошибок.

Тест №3 показал, что игра идет корректно и без ошибок.

1. ОПИСАНИЕ ПРИМЕНЕНИЯ ПРОГРАММЫ
   1. Назначение программы и условия применения

Данная программа используется в качестве развлекательного характера.

Минимальные требования для корректной работы приложения:

- процессор Intel Pentium III 420 Мгц

- ОЗУ 256 Мбайт

- Место на HDD 1024 Мбайт

- VGA или монитор с более высоким разрешением;

- мышь, клавиатура.

Для запуска программы нужно зайти в папку Курсовая работа и запустить файл «Морской Бой». После запуска откроется окно. Пользователь должен выбрать в настройках размерность полей или начать игру, тогда создастся стандартное 10x10.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте была разработана программа «Морской бой».

Было произведено тестирование разработанного приложения. Программа работает корректно при любых действиях пользователя.

Проект может быть доработан следующим образом:

* Создание полей.
* Генерация Кораблей Пользователя
* Клик по полю.

Список использованных источников

1. Google https://www.google.com
2. Youtube https://www.youtube.com
3. wikipedia <https://ru.wikipedia.org/>

Приложение 1

**(обязательное)**

**Листинг программы**

unit Menu;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Imaging.pngimage,

Vcl.ExtCtrls;

type

TForm2 = class(TForm)

Button1: TButton;

Button2: TButton;

Button3: TButton;

procedure FormCreate(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form2: TForm2;

n,n1,SizeImage:integer;

z,z1,z2,z3:Integer;

implementation

uses Field,Option;

{$R \*.dfm}

procedure TForm2.Button1Click(Sender: TObject);

begin

Application.CreateForm(TForm1, Form1);

Form1.Show;

Form2.Visible:=false;

end;

procedure TForm2.Button2Click(Sender: TObject);

begin

Form3.Show;

form2.Visible:=false;

end;

procedure TForm2.Button3Click(Sender: TObject);

begin

halt;

end;

procedure TForm2.FormCreate(Sender: TObject);

begin

SizeImage:=50;

n:=10;

n1:=n;

z:=4;

z1:=3;

z2:=2;

z3:=1;

end;

end.

unit Option;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls;

type

TForm3 = class(TForm)

—охранить: TButton;

RadioGroup1: TRadioGroup;

procedure —охранитьClick(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form3: TForm3;

implementation

{$R \*.dfm}

uses Menu;

procedure TForm3.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

form2.Visible:=true;

Form3.Visible:=false;

end;

procedure TForm3.—охранитьClick(Sender: TObject);

begin

case RadioGroup1.ItemIndex of

0:begin

SizeImage:=50;

n:=10;

n1:=n;

z:=4;

z1:=3;

z2:=2;

z3:=1;

end;

1:begin

SizeImage:=35;

n:=15;

n1:=n;

z:=5;

z1:=4;

z2:=3;

z3:=2;

end;

2:begin

SizeImage:=22;

n:=25;

n1:=n;

z:=6;

z1:=5;

z2:=4;

z3:=3;

end;

end;

form2.Visible:=true;

Form3.Visible:=false;

end;

end.

unit Field;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Grids, Vcl.StdCtrls,Math,GnShip ,

PShip2x2,PShip1x1,MyTypes,Pship3x3,Pship4x4,PImage,Menu,Vcl.Imaging.pngimage,GnTopImageBOT,GnImageUser,GnShipUser,

Vcl.Buttons;

type

MShip1x1 = Array of TShip1;

MShip2x2 = Array of TShip2;

MShip3x3 = Array of TShip3;

MShip4x4 = Array of TShip4;

MImage = Array of Array of TTopImageBOT;

TForm1 = class(TForm)

Panel1: TPanel;

Panel2: TPanel;

Image1: TImage;

BitBtn1: TBitBtn;

procedure FormCreate(Sender: TObject);

procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);

procedure BitBtn1Click(Sender: TObject);

private

{ Private declarations }

public

procedure ImageOnClick(Sender: TObject);

procedure BotImageOnClick(Sender: TObject);

{ Public declarations }

end;

var

Form1: TForm1;

NImage:integer;

Pz,Pz1,Pz2,Pz3:Integer;

DField:MField;

DFieldUser:MField;

Ship1x1:MShip1x1;

Ship2x2:MShip2x2;

Ship3x3:MShip3x3;

Ship4x4:MShip4x4;

PlerShip1x1:MShip1x1;

PlerShip2x2:MShip2x2;

PlerShip3x3:MShip3x3;

PlerShip4x4:MShip4x4;

PlerImage:MImage;

Image:MImage;

implementation

{$R \*.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);

var HT:ThUser;

i,j,k:integer;

begin

HT:=ThUser.Create(False);

HT.Priority:=tpNormal;

for i := 0 to n-1 do

for j := 0 to n-1 do

Image[j,i].Enabled:=True;

end;

procedure TForm1.BotImageOnClick(Sender: TObject);

var Fimage:TTopImageBOT;

i:integer;

begin

if Sender is TTopImageBOT then

begin

Fimage:=TTopImageBOT(Sender);

Fimage.Enabled:=false;

Fimage.Picture.LoadFromFile('Textures\Other\NotKill.png');

for i := 0 to z-1 do

if (Fimage.Fx=PlerShip1x1[i].x) and (Fimage.Fy=PlerShip1x1[i].y) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

/////////////////////////////////////////////////////////////////////

for i := 0 to z1-1 do

begin

if (Fimage.Fx=PlerShip2x2[i].x) and (Fimage.Fy=PlerShip2x2[i].y) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

if (Fimage.Fx=PlerShip2x2[i].x1) and (Fimage.Fy=PlerShip2x2[i].y1) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

end;

/////////////////////////////////////////////////////////////////////////

for i := 0 to z2-1 do

begin

if (Fimage.Fx=PlerShip3x3[i].x) and (Fimage.Fy=PlerShip3x3[i].y) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

if (Fimage.Fx=PlerShip3x3[i].x1) and (Fimage.Fy=PlerShip3x3[i].y1) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

if (Fimage.Fx=PlerShip3x3[i].x2) and (Fimage.Fy=PlerShip3x3[i].y2) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

end;

//////////////////////////////////////////////////////////////////////////

for i := 0 to z3-1 do

begin

if (Fimage.Fx=PlerShip4x4[i].x) and (Fimage.Fy=PlerShip4x4[i].y) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

if (Fimage.Fx=PlerShip4x4[i].x1) and (Fimage.Fy=PlerShip4x4[i].y1) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

if (Fimage.Fx=PlerShip4x4[i].x2) and (Fimage.Fy=PlerShip4x4[i].y2) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

if (Fimage.Fx=PlerShip4x4[i].x3) and (Fimage.Fy=PlerShip4x4[i].y3) then

Fimage.Picture.LoadFromFile('Textures\Other\Kill.png');

end;

end;

end;

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

begin

halt;

end;

procedure TForm1.FormCreate(Sender: TObject);

var i,j:integer;

H:TH;

H1:TGnTopImageBOT;

H2:TGnImageUser;

begin

NImage:=n\*n;

SetLength(Image,NImage);

n:=n+2;

SetLength(DField,n,n);

for i := 0 to n-1 do

for j := 0 to n-1 do

DField[i,j]:=0;

SetLength(DFieldUser,n,n);

for i := 0 to n-1 do

for j := 0 to n-1 do

DFieldUser[i,j]:=0;

H1:=TGnTopImageBOT.Create(False);

H1.Priority:=tpLower;

H2:=TGnImageUser.Create(False);

H2.Priority:=tpNormal;

H:=TH.Create(False);

H.Priority:=tpNormal;

end;

procedure TForm1.ImageOnClick(Sender: TObject);

Var

Fimage:TTopImageBOT;

i,a,b:integer;

begin

if Sender is TTopImageBOT then

begin

Fimage:=TTopImageBOT(Sender);

Fimage.Enabled:=false;

Fimage.Picture.LoadFromFile('Textures/Other/Empty.png');

for i := 0 to z-1 do

if (Fimage.Fx=Ship1x1[i].x) and (Fimage.Fy=Ship1x1[i].y) then

Fimage.Picture:=Ship1x1[i].Ship1Image.Picture;

/////////////////////////////////////////////////////////////////////

for i := 0 to z1-1 do

begin

if (Fimage.Fx=Ship2x2[i].x) and (Fimage.Fy=Ship2x2[i].y) then

Fimage.Picture:=Ship2x2[i].Ship1Image.Picture;

if (Fimage.Fx=Ship2x2[i].x1) and (Fimage.Fy=Ship2x2[i].y1) then

Fimage.Picture:=Ship2x2[i].Ship2Image.Picture;

end;

/////////////////////////////////////////////////////////////////////////

for i := 0 to z2-1 do

begin

if (Fimage.Fx=Ship3x3[i].x) and (Fimage.Fy=Ship3x3[i].y) then

Fimage.Picture:=Ship3x3[i].Ship1Image.Picture;

if (Fimage.Fx=Ship3x3[i].x1) and (Fimage.Fy=Ship3x3[i].y1) then

Fimage.Picture:=Ship3x3[i].Ship2Image.Picture;

if (Fimage.Fx=Ship3x3[i].x2) and (Fimage.Fy=Ship3x3[i].y2) then

Fimage.Picture:=Ship3x3[i].Ship3Image.Picture;

end;

//////////////////////////////////////////////////////////////////////////

for i := 0 to z3-1 do

begin

if (Fimage.Fx=Ship4x4[i].x) and (Fimage.Fy=Ship4x4[i].y) then

Fimage.Picture:=Ship4x4[i].Ship1Image.Picture;

if (Fimage.Fx=Ship4x4[i].x1) and (Fimage.Fy=Ship4x4[i].y1) then

Fimage.Picture:=Ship4x4[i].Ship2Image.Picture;

if (Fimage.Fx=Ship4x4[i].x2) and (Fimage.Fy=Ship4x4[i].y2) then

Fimage.Picture:=Ship4x4[i].Ship3Image.Picture;

if (Fimage.Fx=Ship4x4[i].x3) and (Fimage.Fy=Ship4x4[i].y3) then

Fimage.Picture:=Ship4x4[i].Ship4Image.Picture;

end;

while True do

begin

a:=RandomRange(1,n-1);

b:=RandomRange(1,n-1);

if PlerImage[a,b].Enabled then

begin

BotImageOnClick(PlerImage[a,b]);

exit;

end;

end;

end;

end;

end.

unit GnImageUser;

interface

uses

System.Classes;

type

TGnImageUser = class(TThread)

private

FIimage:integer;

procedure CreateGnTopImageBOT(LeftShift,TopShift,j,i:integer);

protected

procedure Execute; override;

end;

implementation

uses Field,PShip1x1,PShip2x2,Pship3x3,Pship4x4,PImage,System.SysUtils,Menu;

procedure TGnImageUser.CreateGnTopImageBOT(LeftShift, TopShift, j, i: integer);

begin

if (j<>0) or (i<>n-1) or (i<>0) or (j<>n-1) then

PlerImage[j,i]:=TTopImageBOT.Create(PlerImage[j,i]);

if (j<>0) and (i<>n-1) and (i<>0) and (j<>n-1) then

begin

PlerImage[j,i]:=TTopImageBOT.Create(Form1.Panel1);

PlerImage[j,i].Parent:=Form1.Panel1;

PlerImage[j,i].Left:=0+LeftShift;

PlerImage[j,i].Top:=0+TopShift;

PlerImage[j,i].Width:=SizeImage;

PlerImage[j,i].Height:=SizeImage;

PlerImage[j,i].Picture.LoadFromFile('Textures/Other/Empty.png');

PlerImage[j,i].Stretch:=True;

PlerImage[j,i].Visible:=true;

PlerImage[j,i].Tag:=i;

PlerImage[j,i].Enabled:=true;

end;

end;

procedure TGnImageUser.Execute;

var i,j,LeftShift,TopShift:integer;

begin

SetLength(PlerImage,n,n);

LeftShift:=0;

TopShift:=0;

for i := 0 to n-1 do

begin

for j := 0 to n-1 do

begin

CreateGnTopImageBOT(LeftShift,TopShift,j,i);

PlerImage[j,i].Fx:=j;

PlerImage[j,i].Fy:=i;

LeftShift:=LeftShift+SizeImage;

end;

LeftShift:=0;

TopShift:=TopShift+SizeImage;

end;

end;

end.

unit GnShip;

interface

uses

System.Classes;

type

TH = class(TThread)

private

{ Private declarations }

protected

procedure Execute; override;

end;

implementation

uses Field,PShip1x1,PShip2x2,Pship3x3,Pship4x4,Menu;

procedure TH.Execute;

var i:integer;

begin

SetLength(Ship4x4,z3);

for I := 0 to z3-1 do

Ship4x4[i]:=TShip4.CreatBOT(N-2,DField);

SetLength(Ship3x3,z2);

for I := 0 to z2-1 do

Ship3x3[i]:=TShip3.CreatBOT(N-2,DField);

SetLength(Ship2x2,z1);

for i := 0 to z1-1 do

Ship2x2[i]:=TShip2.CreatBOT(N-2,DField);

SetLength(Ship1x1,z);

for I := 0 to z-1 do

begin

Ship1x1[i]:=TShip1.CreatBOT(N-2,DField);

end;

end;

end.

unit GnShipUser;

interface

uses

System.Classes;

type

ThUser = class(TThread)

private

{ Private declarations }

protected

procedure Execute; override;

public

end;

implementation

uses Field,PShip1x1,PShip2x2,Pship3x3,Pship4x4,Menu;

procedure ThUser.Execute;

var i,j,k:integer;

begin

SetLength(PlerShip4x4,z3);

for I := 0 to z3-1 do

PlerShip4x4[i]:=TShip4.CreatBOT(N-2,DFieldUser);

SetLength(PlerShip3x3,z2);

for I := 0 to z2-1 do

PlerShip3x3[i]:=TShip3.CreatBOT(N-2,DFieldUser);

SetLength(PlerShip2x2,z1);

for i := 0 to z1-1 do

PlerShip2x2[i]:=TShip2.CreatBOT(N-2,DFieldUser);

SetLength(PlerShip1x1,z);

for I := 0 to z-1 do

begin

PlerShip1x1[i]:=TShip1.CreatBOT(N-2,DFieldUser);

end;

for i := 1 to n-2 do

for j := 1 to n-2 do

begin

for k := 0 to z-1 do

if (PlerImage[j,i].Fx=PlerShip1x1[k].x) and (PlerImage[j,i].Fy= PlerShip1x1[k].y) then

PlerImage[j,i].Picture:=PlerShip1x1[k].Ship1Image.Picture;

for k := 0 to z1-1 do

begin

if (PlerImage[j,i].Fx=PlerShip2x2[k].x) and (PlerImage[j,i].Fy= PlerShip2x2[k].y) then

PlerImage[j,i].Picture:=PlerShip2x2[k].Ship1Image.Picture;

if (PlerImage[j,i].Fx=PlerShip2x2[k].x1) and (PlerImage[j,i].Fy= PlerShip2x2[k].y1) then

PlerImage[j,i].Picture:=PlerShip2x2[k].Ship2Image.Picture;

end;

for k := 0 to z2-1 do

begin

if (PlerImage[j,i].Fx=PlerShip3x3[k].x) and (PlerImage[j,i].Fy= PlerShip3x3[k].y) then

PlerImage[j,i].Picture:=PlerShip2x2[k].Ship1Image.Picture;

if (PlerImage[j,i].Fx=PlerShip3x3[k].x1) and (PlerImage[j,i].Fy= PlerShip3x3[k].y1) then

PlerImage[j,i].Picture:=PlerShip3x3[k].Ship2Image.Picture;

if (PlerImage[j,i].Fx=PlerShip3x3[k].x2) and (PlerImage[j,i].Fy= PlerShip3x3[k].y2) then

PlerImage[j,i].Picture:=PlerShip3x3[k].Ship3Image.Picture;

end;

for k := 0 to z3-1 do

begin

if (PlerImage[j,i].Fx=PlerShip4x4[k].x) and (PlerImage[j,i].Fy= PlerShip4x4[k].y) then

PlerImage[j,i].Picture:=PlerShip4x4[k].Ship1Image.Picture;

if (PlerImage[j,i].Fx=PlerShip4x4[k].x1) and (PlerImage[j,i].Fy= PlerShip4x4[k].y1) then

PlerImage[j,i].Picture:=PlerShip4x4[k].Ship2Image.Picture;

if (PlerImage[j,i].Fx=PlerShip4x4[k].x2) and (PlerImage[j,i].Fy= PlerShip4x4[k].y2) then

PlerImage[j,i].Picture:=PlerShip4x4[k].Ship3Image.Picture;

if (PlerImage[j,i].Fx=PlerShip4x4[k].x3) and (PlerImage[j,i].Fy= PlerShip4x4[k].y3) then

PlerImage[j,i].Picture:=PlerShip4x4[k].Ship4Image.Picture;

end;

end;

Form1.BitBtn1.Visible:=false;

end;

end.

unit GnTopImageBOT;

interface

uses

System.Classes;

type

TGnTopImageBOT = class(TThread)

private

FIimage:integer;

procedure CreateGnTopImageBOT(LeftShift,TopShift,j,i:integer);

{ Private declarations }

protected

procedure Execute; override;

end;

implementation

uses Field,PShip1x1,PShip2x2,Pship3x3,Pship4x4,PImage,System.SysUtils,Menu;

procedure TGnTopImageBOT.CreateGnTopImageBOT(LeftShift,TopShift,j,i:integer);

begin

if (j<>0) or (i<>n-1) or (i<>0) or (j<>n-1) then

Image[j,i]:=TTopImageBOT.Create(Image[j,i]);

if (j<>0) and (i<>n-1) and (i<>0) and (j<>n-1) then

begin

Image[j,i]:=TTopImageBOT.Create(Form1.Panel2);

Image[j,i].Parent:=Form1.Panel2;

Image[j,i].Left:=0+LeftShift;

Image[j,i].Top:=0+TopShift;

Image[j,i].Width:=SizeImage;

Image[j,i].Height:=SizeImage;

Image[j,i].Picture.LoadFromFile('Textures/Other/TopImage.png');

Image[j,i].Stretch:=True;

Image[j,i].Visible:=true;

Image[j,i].Tag:=i;

Image[j,i].OnClick:=Form1.ImageOnClick;

end;

end;

procedure TGnTopImageBOT.Execute;

var i,j,LeftShift,TopShift:integer;

begin

SetLength(Image,n,n);

LeftShift:=0;

TopShift:=0;

for i := 0 to n-1 do

begin

for j := 0 to n-1 do

begin

CreateGnTopImageBOT(LeftShift,TopShift,j,i);

Image[j,i].Fx:=j;

Image[j,i].Fy:=i;

LeftShift:=LeftShift+SizeImage;

end;

LeftShift:=0;

TopShift:=TopShift+SizeImage;

end;

for i := 0 to n-1 do

for j := 0 to n-1 do

Image[j,i].Enabled:=false;

end;

end.

unit MyTypes;

interface

uses System.SysUtils;

type

TArray = Array of Integer;

MField = Array of Array of byte;

Mas = Array of integer;

implementation

end.

unit PImage;

interface

uses Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.Grids, Vcl.StdCtrls,Math,GnShip ,

PShip2x2,PShip1x1,MyTypes,Pship3x3,Pship4x4, Vcl.Imaging.pngimage;

type

TTopImageBOT = class (TImage)

Image1:TImage;

public

Fx,Fy:integer;

end;

implementation

{ TTopImage }

end.

program Project1;

uses

Vcl.Forms,

Field in 'Field.pas' {Form1},

PShip1x1 in 'PShip1x1.pas',

PShip2x2 in 'PShip2x2.pas',

GnShip in 'GnShip.pas',

MyTypes in 'MyTypes.pas',

Pship3x3 in 'Pship3x3.pas',

Pship4x4 in 'Pship4x4.pas',

GnTopImageBOT in 'GnTopImageBOT.pas',

GnShipUser in 'GnShipUser.pas',

Menu in 'Menu.pas' {Form2},

Option in 'Option.pas' {Form3};

{$R \*.res}

begin

Application.Initialize;

Application.MainFormOnTaskbar := True;

Application.CreateForm(TForm2, Form2);

Application.CreateForm(TForm3, Form3);

Application.Run;

end.

unit PShip1x1;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Math,mytypes,Vcl.Imaging.pngimage;

type

TShip1 = class

private

procedure Image1;

function RandomOnCentre(a,b:integer; DField: MField):boolean;

public

Ship1Image: TImage;

x,y:integer;

Live1:byte;

constructor CreatBOT( NewN:integer; var DField:MField);

end;

implementation

constructor TShip1.CreatBOT(NewN:integer; var DField:MField);

var i,a,b,Z:integer;

begin

Image1;

while true do

begin

a:=RandomRange(1,NewN+1);

b:=RandomRange(1,NewN+1);

if RandomOnCentre(a,b,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

exit;

end;

end;

{ TShip1 }

end;

procedure TShip1.Image1;

begin

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

end;

function TShip1.RandomOnCentre(a, b: integer; DField: MField): boolean;

begin

Result:=False;

if (DField[a,b]=0) then

if (DField[a+1,b]<>1) then

if (DField[a-1,b]<>1) then

if (DField[a,b+1]<>1) then

if (DField[a,b-1]<>1) then

if (DField[a+1,b+1]<>1) then

if (DField[a+1,b-1]<>1) then

if (DField[a-1,b+1]<>1) then

if (DField[a-1,b-1]<>1) then

Result:=true;

end;

end.

unit PShip2x2;

interface

uses

MyTypes,Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Math, Vcl.Imaging.pngimage;

type

TShip2 = class

private

procedure Image2;

procedure TopDownImage(b, b1: integer);

protected

procedure qSort(var A:mas; min, max: Integer);

function RandomOnCentre(a,b,N:integer; DField: MField):boolean;

function RandomOnLeft(a,b,N:integer; DField: MField):boolean;

function RandomOnTop(a,b,N:integer; DField: MField):boolean;

function RandomOnRight(a,b,N:integer; DField: MField):boolean;

function RandomOnDown(a,b,N:integer; DField: MField):boolean;

public

Ship1Image: TImage;

Ship2Image: TImage;

x,y:integer;

x1,y1:integer;

Live1:byte;

Live2:byte;

constructor CreatBOT( NewN:integer; var DField:MField);

end;

implementation

{ TShip1 }

constructor TShip2.CreatBOT(NewN: integer; var DField: MField);

var i,a,b:integer; Ship1x2:byte;

begin

Image2;

while True do

begin

a:=RandomRange(1,NewN+1);

b:=RandomRange(1,NewN+1);

if RandomOnCentre(a,b,NewN,DField) then

begin

Ship1x2:=RandomRange(1,5);

case Ship1x2 of

1:begin

if RandomOnLeft(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a-1,b]:=1;

Live2:=1;

x1:=a-1;

y1:=b;

exit;

end;

end;

2:begin

if RandomOnTop(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b-1]:=1;

Live2:=1;

x1:=a;

y1:=b-1;

exit;

end;

end;

3:begin

if RandomOnDown(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b+1]:=1;

Live2:=1;

x1:=a;

y1:=b+1;

exit;

end;

end;

4:begin

if RandomOnRight(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a+1,b]:=1;

Live2:=1;

x1:=a+1;

y1:=b;

exit;

end;

end;

end;

end;

end;

end;

procedure TShip2.Image2;

begin

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

Ship2Image:=TImage.Create(Ship2Image);

Ship2Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

end;

procedure TShip2.qSort(var A:mas; min, max: Integer);

var i, j, supp, tmp: Integer;

begin

supp:=A[max-((max-min) div 2)];

i:=min; j:=max;

while i<j do

begin

while A[i]<supp do i:=i+1;

while A[j]>supp do j:=j-1;

if i<=j then

begin

tmp:=A[i]; A[i]:=A[j]; A[j]:=tmp;

i:=i+1; j:=j-1;

end;

end;

if min<j then qSort(A, min, j);

if i<max then qSort(A, i, max);

end;

function TShip2.RandomOnCentre(a, b, N: integer; DField: MField): boolean;

begin

Result:=false;

if (DField[a,b]=0) then

if (DField[a+1,b]<>1) then

if (DField[a-1,b]<>1) then

if (DField[a,b+1]<>1) then

if (DField[a,b-1]<>1) then

if (DField[a+1,b+1]<>1) then

if (DField[a+1,b-1]<>1) then

if (DField[a-1,b+1]<>1) then

if (DField[a-1,b-1]<>1) then

Result:=true;

end;

function TShip2.RandomOnLeft(a, b, N: integer; DField: MField): boolean;

begin

result:=false;

if ((a-1)<>0) then

if (DField[a-2,b]<>1) then

if (DField[a-2,b+1]<>1) then

if (DField[a-2,b-1]<>1) then

result:=true;

end;

function TShip2.RandomOnDown(a, b, N: integer; DField: MField): boolean;

begin

Result:=false;

if (b+1)<>(N+1) then

if (DField[a,b+2]<>1) then

if (DField[a-1,b+2]<>1) then

if (DField[a+1,b+2]<>1) then

Result:=true;

end;

function TShip2.RandomOnRight(a, b, N: integer; DField: MField): boolean;

begin

Result:=false;

if (a+1)<>(N+1) then

if (DField[a+2,b]<>1) then

if (DField[a+2,b+1]<>1) then

if (DField[a+2,b-1]<>1) then

Result:=true;

end;

function TShip2.RandomOnTop(a, b,N: integer; DField: MField): boolean;

begin

Result:=false;

if (b-1)<>0 then

if (DField[a,b-2]<>1) then

if (DField[a-1,b-2]<>1) then

if (DField[a+1,b-2]<>1) then

Result:=true;

end;

procedure TShip2.TopDownImage(b, b1: integer);

var

TD:integer;

begin

if b>b1 then

begin

TD:=RandomRange(1,3);

case TD of

1:begin

Ship2Image:=TImage.Create(Ship2Image);

Ship2Image.Picture.LoadFromFile('Textures/Ship2x2/DownShip\_Down.png');

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship2x2/TopShip\_Top.png');

end;

2:begin

Ship2Image:=TImage.Create(Ship2Image);

Ship2Image.Picture.LoadFromFile('Textures/Ship2x2/TopShip\_Down.png');

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship2x2/DownShip\_Top.png');

end;

end;

end;

if b<b1 then

begin

TD:=RandomRange(1,3);

case TD of

1:begin

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship2x2/DownShip\_Down.png');

Ship2Image:=TImage.Create(Ship2Image);

Ship2Image.Picture.LoadFromFile('Textures/Ship2x2/TopShip\_Top.png');

end;

2:begin

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship2x2/TopShip\_Down.png');

Ship2Image:=TImage.Create(Ship2Image);

Ship2Image.Picture.LoadFromFile('Textures/Ship2x2/DownShip\_Top.png');

end;

end;

end;

end;

end.

unit Pship3x3;

interface

uses

MyTypes,Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Math,PShip2x2;

type

TShip3 = class (TShip2)

private

procedure Image3;

public

Ship3Image: TImage;

Live3:byte;

x2,y2:integer;

constructor CreatBOT( NewN:integer; var DField:MField);

end;

implementation

{ TShip3 }

procedure TShip3.Image3;

begin

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

Ship2Image:=TImage.Create(Ship2Image);

Ship2Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

Ship3Image:=TImage.Create(Ship3Image);

Ship3Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

end;

{ TShip3 }

constructor TShip3.CreatBOT(NewN: integer; var DField: MField);

var i,a,b:integer; Ship1x2,Ship1x3:byte;

begin

Image3;

while True do

begin

a:=RandomRange(1,NewN+1);

b:=RandomRange(1,NewN+1);

if RandomOnCentre(a,b,NewN,DField) then

begin

Ship1x2:=RandomRange(1,5);

case Ship1x2 of

1:begin

if RandomOnLeft(a,b,NewN,DField) then

begin

Ship1x3:=RandomRange(1,3);

case Ship1x3 of

1:begin

if RandomOnLeft(a-1,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a-1,b]:=1;

Live2:=1;

x1:=a-1;

y1:=b;

DField[a-2,b]:=1;

Live3:=1;

x2:=a-2;

y2:=b;

exit;

end;

end;

2:begin

if RandomOnRight(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a-1,b]:=1;

Live2:=1;

x1:=a-1;

y1:=b;

DField[a+1,b]:=1;

Live3:=1;

x2:=a+1;

y2:=b;

exit;

end;

end;

end;

end;

end;

2:begin

if RandomOnRight(a,b,NewN,DField) then

begin

Ship1x3:=RandomRange(1,3);

case Ship1x3 of

1:begin

if RandomOnLeft(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a+1,b]:=1;

Live2:=1;

x1:=a+1;

y1:=b;

DField[a-1,b]:=1;

Live3:=1;

x2:=a-1;

y2:=b;

exit;

end;

end;

2:begin

if RandomOnRight(a+1,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a+1,b]:=1;

Live2:=1;

x1:=a+1;

y1:=b;

DField[a+2,b]:=1;

Live3:=1;

x2:=a+2;

y2:=b;

exit;

end;

end;

end;

end;

end;

3: begin

if RandomOnTop(a,b,NewN,DField) then

begin

Ship1x3:=RandomRange(1,3);

case Ship1x3 of

1:begin

if RandomOnTop(a,b-1,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b-1]:=1;

Live2:=1;

x1:=a;

y1:=b-1;

DField[a,b-2]:=1;

Live3:=1;

x2:=a;

y2:=b-2;

exit;

end;

end;

2:begin

if RandomOnDown(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b+1]:=1;

Live2:=1;

x1:=a;

y1:=b+1;

DField[a,b-1]:=1;

Live3:=1;

x2:=a;

y2:=b-1;

exit;

end;

end;

end;

end;

end;

4: begin

if RandomOnDown(a,b,NewN,DField) then

begin

Ship1x3:=2;

case Ship1x3 of

1:begin

if RandomOnDown(a,b+1,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b+1]:=1;

Live2:=1;

x1:=a;

y1:=b+1;

DField[a,b+2]:=1;

Live3:=1;

x2:=a;

y2:=b+2;

exit;

end;

end;

2:begin

if RandomOnTop(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b+1]:=1;

Live2:=1;

x1:=a;

y1:=b+1;

DField[a,b-1]:=1;

Live3:=1;

x2:=a;

y2:=b-1;

exit;

end;

end;

end;

end;

end;

end;

end;

end;

end;

end.

unit Pship4x4;

interface

uses

MyTypes,Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Math,PShip2x2;

Type

TShip4 = class (TShip2)

private

procedure Image4;

public

Ship3Image: TImage;

Live3:byte;

x2,y2:integer;

Ship4Image: TImage;

Live4:byte;

x3,y3:integer;

constructor CreatBOT( NewN:integer; var DField:MField);

end;

implementation

{ TShip3 }

constructor TShip4.CreatBOT(NewN: integer; var DField: MField);

var i,a,b:integer; Ship1x2,Ship1x3,Ship1x4:byte;

begin

Image4;

while True do

begin

a:=RandomRange(1,NewN+1);

b:=RandomRange(1,NewN+1);

if RandomOnCentre(a,b,NewN,DField) then

begin

Ship1x2:=RandomRange(1,3);

case Ship1x2 of

1:begin

if RandomOnLeft(a,b,NewN,DField) then

begin

Ship1x3:=RandomRange(1,3);

case Ship1x3 of

1:begin

if RandomOnLeft(a-1,b,NewN,DField) then

begin

Ship1x4:=RandomRange(1,3);

case Ship1x4 of

1:begin

if RandomOnLeft(a-2,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a-1,b]:=1;

Live2:=1;

x1:=a-1;

y1:=b;

DField[a-2,b]:=1;

Live3:=1;

x2:=a-2;

y2:=b;

DField[a-3,b]:=1;

Live4:=1;

x3:=a-3;

y3:=b;

exit;

end;

end;

2:begin

if RandomOnRight(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a-1,b]:=1;

Live2:=1;

x1:=a-1;

y1:=b;

DField[a-2,b]:=1;

Live3:=1;

x2:=a-2;

y2:=b;

DField[a+1,b]:=1;

Live4:=1;

x3:=a+1;

y3:=b;

exit;

end;

end;

end;

end;

end;

2:begin

if RandomOnRight(a,b,NewN,DField) then

begin

Ship1x4:=RandomRange(1,3);

case Ship1x4 of

1:begin

if RandomOnRight(a+1,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a-1,b]:=1;

Live2:=1;

x1:=a-1;

y1:=b;

DField[a+1,b]:=1;

Live3:=1;

x2:=a+1;

y2:=b;

DField[a+2,b]:=1;

Live4:=1;

x3:=a+2;

y3:=b;

exit;

end;

end;

2:begin

if RandomOnLeft(a-1,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a-1,b]:=1;

Live2:=1;

x1:=a-1;

y1:=b;

DField[a+1,b]:=1;

Live3:=1;

x2:=a+1;

y2:=b;

DField[a-2,b]:=1;

Live4:=1;

x3:=a-2;

y3:=b;

exit;

end;

end;

end;

end;

end;

end;

end;

end;

///////////////////////////////////////////////////////////////////////////

2: begin

if RandomOnTop(a,b,NewN,DField) then

begin

Ship1x3:=2;//RandomRange(1,3);

case Ship1x3 of

1:begin

if RandomOnTop(a,b-1,NewN,DField) then

begin

Ship1x4:=RandomRange(1,3);

case Ship1x4 of

1:begin

if RandomOnTop(a,b-2,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b-1]:=1;

Live2:=1;

x1:=a;

y1:=b-1;

DField[a,b-2]:=1;

Live3:=1;

x2:=a;

y2:=b-2;

DField[a,b-3]:=1;

Live4:=1;

x3:=a;

y3:=b-3;

exit;

end;

end;

2:begin

if RandomOnDown(a,b,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b-1]:=1;

Live2:=1;

x1:=a;

y1:=b-1;

DField[a,b-2]:=1;

Live3:=1;

x2:=a;

y2:=b-2;

DField[a,b+1]:=1;

Live4:=1;

x3:=a;

y3:=b+1;

exit;

end;

end;

end;

end;

end;

2:begin

if RandomOnDown(a,b,NewN,DField) then

begin

Ship1x4:=RandomRange(1,3);

case Ship1x4 of

1:begin

if RandomOnDown(a,b+1,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b+1]:=1;

Live2:=1;

x1:=a;

y1:=b+1;

DField[a,b-1]:=1;

Live3:=1;

x2:=a;

y2:=b-1;

DField[a,b+2]:=1;

Live4:=1;

x3:=a;

y3:=b+2;

exit;

end;

end;

2:begin

if RandomOnTop(a,b-1,NewN,DField) then

begin

DField[a,b]:=1;

Live1:=1;

x:=a;

y:=b;

DField[a,b+1]:=1;

Live2:=1;

x1:=a;

y1:=b+1;

DField[a,b-1]:=1;

Live3:=1;

x2:=a;

y2:=b-1;

DField[a,b-2]:=1;

Live4:=1;

x3:=a;

y3:=b-2;

exit;

end;

end;

end;

end;

end;

end;

end;

end;

end;

end;

end;

end;

procedure TShip4.Image4;

begin

Ship1Image:=TImage.Create(Ship1Image);

Ship1Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

Ship2Image:=TImage.Create(Ship2Image);

Ship2Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

Ship3Image:=TImage.Create(Ship3Image);

Ship3Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

Ship4Image:=TImage.Create(Ship4Image);

Ship4Image.Picture.LoadFromFile('Textures/Ship3x3/Center.png');

end;

end.