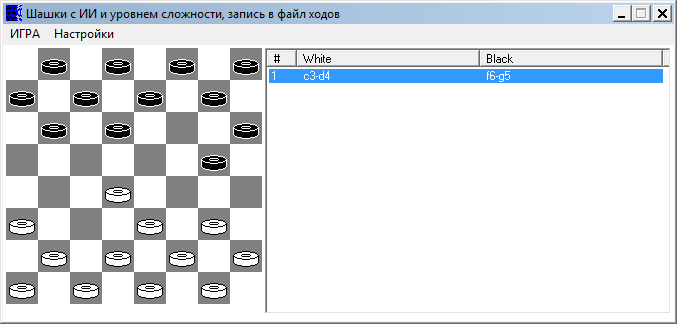
**Скриншоты программы**

****

**Исходный код программы**

unit MainUnit;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,

PosFrame, StdCtrls, GameLogic, Menus, ComCtrls, Clipbrd, ActnList;

// исплдьзование библиотек

const

MM\_DOMOVE = WM\_USER + 1; // бить на 1

MM\_DEBUG = WM\_USER + 2;//2

MM\_IS\_ANIMATION = WM\_USER + 3;//3 хода

type

TMode = (mdMachineWhite, mdMachineBlack, mdTwoMachine, mdView);// режим игры, в виде типа

TGameHistory = class // класс история игры, с вызовом соответствующих функций

private

function GetPartyView: TListView; // загрузка стартового вида партии

function GetPositionFrame: TPositionFrame;{позиций aboueh на доске }

private

FPositions: array[0..255] of TPosition;{массив позиций}

FMoveNo: Integer; {индикатор невозможности хода(Номер хода)}

procedure AddBlackMove(const Move: string); // ход белых(добавление)

procedure AddWhiteMove(const Move: string);// ход черных(добавление в историю )

property PositionFrame: TPositionFrame read GetPositionFrame;{совмещение вида доскии фигур }

property PartyView: TListView read GetPartyView;{итоговый вид партиии }

public

procedure NewGame;{инициализация новой игры}

procedure AddMove(NewPosition: TPosition);{хода}

procedure Undo;{возврат хода }

property MoveNo: Integer read FMoveNo write FMoveNo;{Номер хода }

end;

TMainForm = class(TForm)// класс инициализации оболочки игры с вызовом соответствующих ф-ций

PositionFrame: TPositionFrame;

MainMenu: TMainMenu;

GameMenu: TMenuItem;

NewItem: TMenuItem;

Separator1: TMenuItem;

BeginerItem: TMenuItem;

IntermediateItem: TMenuItem;

ExpertItem: TMenuItem;

Separator2: TMenuItem;

ExitItem: TMenuItem;

ModeMenu: TMenuItem;

MachineWhiteItem: TMenuItem;

MachineBlackItem: TMenuItem;

TwoMachineItem: TMenuItem;

ViewItem: TMenuItem;

Separator3: TMenuItem;

FlipBoardItem: TMenuItem;

PartyView: TListView;

CopyGameItem: TMenuItem;

Separator4: TMenuItem;

UndoMoveItem: TMenuItem;

ActionList: TActionList;

NewGameAction: TAction;

BeginerAction: TAction;

IntermediateAction: TAction;

ExpertAction: TAction;

UndoMoveAction: TAction;

ExitAction: TAction;

MachineWhiteAction: TAction;

MachineBlackAction: TAction;

TwoMachineAction: TAction;

ViewGameAction: TAction;

FlipBoardAction: TAction;

SetPositionAction: TAction;

AddToLibraryAction: TAction;

CopyGameAction: TAction;

procedure FormShow(Sender: TObject);// показ формы

procedure SelectCellBtnClick(Sender: TObject);// обработка нажатий мышкой

procedure FormResize(Sender: TObject);// изменение размеров формы

procedure FormCreate(Sender: TObject); // создание формы

procedure FormDestroy(Sender: TObject);// удаление

procedure UndoMoveItemClick(Sender: TObject);// отмена хода(обработчик пункта меню)

procedure NewGameActionExecute(Sender: TObject);// нов.игра

procedure LevelActionExecute(Sender: TObject);// выбор уровня играющего

procedure UndoMoveActionExecute(Sender: TObject); // обработка действия в игре(графически и //логиччески )

procedure ActionListUpdate(Action: TBasicAction; var Handled: Boolean);// прорисовка действий

procedure ExitActionExecute(Sender: TObject);// выход из игры

procedure MachineWhiteActionExecute(Sender: TObject);// комп белыми

procedure MachineBlackActionExecute(Sender: TObject);// комп чрными

procedure TwoMachineActionExecute(Sender: TObject);// комп сам с собой

procedure ViewGameActionExecute(Sender: TObject);// два игрока без ии

procedure FlipBoardActionExecute(Sender: TObject);// переворот доски

procedure CopyGameActionExecute(Sender: TObject);// партию в файл

private

FDeep: Integer;// глубина (сложностть(числовой показатель))

FGameHistory: TGameHistory;{инициализация истории}

FMode: TMode;// переход по режимам

FThreadHandle: THandle;// положение переключателя(легкий, средний, по уровню и т.д)

procedure AcceptMove(Sender: TObject; const NewPosition: TPosition);// перебор разрешенных ходов

procedure TuneState;// выбор режима настроек

procedure StopThinking;{данная процедура заставляетт И.И выйти из «раздумья», когда человек делает ход}

procedure DoMove(var Message: TMessage); message MM\_DOMOVE;// ход

procedure DoDebug(var Message: TMessage); message MM\_DEBUG;// прогноз хода

procedure IsAnimation(var Message: TMessage); message MM\_IS\_ANIMATION; // отработка анимаци

property Mode: TMode read FMode;// режим игры

property Deep: Integer read FDeep write FDeep;// глубина мышления ии(сложность )

property ThreadHandle: THandle read FThreadHandle write FThreadHandle; // получение потока ходов

property GameHistory: TGameHistory read FGameHistory write FGameHistory;// история игоы

procedure Deselect(Action: TAction; const Category: string);// перебор категориий в меню

end;

var

MainForm: TMainForm;

implementation

uses GameTactics;

{$R \*.DFM}

function Thinker(APosition: Pointer): Integer;{обдумывание позиции }

var

Position: TPosition; //получаем позицию

Estimate: Integer; //фактор оценивания(к примеру, от 1 до 5 )

begin

Position := TPosition(APosition^); //при этом, позиция –указатель нна сооотв. тип

SelectMove(Position, MainForm.Deep, Estimate);{запускаем процедуру перебора ходов, на вход получив: позицию, сложность с которой ИИ должен думать и оценку (параметр, который должен получить на выходе, )}

SendMessage(MainForm.Handle, MM\_DOMOVE, Integer(@Position), Estimate);// в соответствии с решением //ии передает параметр уже процедуре MM\_DOMOVE, Integer(на входе – выбранная позиция и начальная //оценка)

Result := 0;// возврат результата ф-ции

end;

procedure TMainForm.FormShow(Sender: TObject);

var

Position: TPosition; {при показе формы, получаем позицию, чтоб прорисовать ее на доске }

begin

LoadLib; {подключение модулей, просчитывающих ходы(gameTacticts)}

Position := StartBoard;// позицию делаем стартовой(для игры)

PositionFrame.OnAcceptMove := AcceptMove;//просчитываем, есть ли оцененны ход в списке //разрешенных(если да, вызываем данный метод, и.и готов сделать верный ход )

NewGameAction.Execute;// запускаем новую игру

BeginerAction.Execute;// для игрока, ранга «начинающий»

MachineBlackAction.Execute;// при этом, машина игрпает черными, человек – белыми(вызываем соотв. Метод)

end;

procedure TMainForm.SelectCellBtnClick(Sender: TObject);

begin

PositionFrame.SelectCell(1, 6);{обработка нажатий на доске, сформированой методом positionFrame}

end;

procedure TMainForm.AcceptMove(Sender: TObject; const NewPosition: TPosition);

var

St: string;// ведем историю ходов(разрешенных), записывая в данную переменную

begin

GameHistory.AddMove(NewPosition); {методом addmove вносим позицию на проверку,, в список «разрешенных»}

PositionFrame.SetPosition(NewPosition);{проверяем на игровом поле, может ли существовать позиция}

St := GameOver(NewPosition);{флагом проверяем, может ли быть продолжена игра}

if St <> ' ' then{если игра может быть продолжена, но, ход не является разрешенным, – ход не равен пустоте, например – фигура при этом ходит буквой Г }

begin

ShowMessage(St);// предупреждаемм, что такого хода не сделать

PositionFrame.AcceptMove := False;// выносиим и выводим результат – данного хода сделано не будет, так //как ход делается тлько в том случае, когда AcceptMove вернет true

Exit;// завершаем операцию

end;

TuneState;// процедура настройки ууровня игры, делает игровой процесс более сообразным критериям //пользователя.

end;

procedure TMainForm.FormResize(Sender: TObject);

begin

{прорисовка доски и клеток на нужном друг ри друга расстоянии и правильного цвета\ в правильном количестве}

PositionFrame.Left := 3;

PositionFrame.Top := 3;

PartyView.Left := PositionFrame.Left + PositionFrame.Width + 3;

PartyView.Width := ClientWidth - PositionFrame.Width - 9;

PartyView.Top := 3;

PartyView.Height := PositionFrame.Height;

PartyView.Columns[0].Width := 30;

PartyView.Columns[1].Width := (PartyView.Width - 40) div 2;

PartyView.Columns[2].Width := (PartyView.Width - 40) div 2;

end;

procedure TMainForm.DoMove(var Message: TMessage);

var

{процедура хода}

NewPosition: TPosition;// получаем переменнную, тима позиция

begin

NewPosition := TPosition(Pointer(Message.WParam)^);// точнее – указатель на не

CloseHandle(ThreadHandle);// проверяем ближайшие возможные ходы

ThreadHandle := 0;// проверенный ход(по умолчанию равен 0, если ход проверен и «одобрен», флаг меняется)

AcceptMove(nil, NewPosition);// проверяем ход на разрежённость

end;

procedure TMainForm.FormCreate(Sender: TObject);

begin

FMode := mdMachineBlack;// устанавливаем начальную позицию (человек белыми – комп чепными )

DoubleBuffered := True;// используем двойную буферизацию( для улучшения кач-вва отображения)

FGameHistory := TGameHistory.Create;//создаем список с историей ходов

end;

procedure TMainForm.TuneState;

var

RunThinker: Boolean; // флговая переменная для проверки того, запущено ли «обдумывание» ходов //пользователя в определенном им режиме

ThreadId: Cardinal; // инициализируем индикатор потока, для нужд ИИ

Index: Integer;

V: Integer;

begin

if ThreadHandle <> 0 then StopThinking;// если идентификатор потока не равен остановочному, то есть ИИ //должен думать над ходом

PositionFrame.AcceptMove := (Mode = mdView)// то, ход ищется среди разрешенных

or ((Mode = mdMachineWhite) and (PositionFrame.Position.Active = ActiveBlack)) // при условии, что //человек играет черными

or ((Mode = mdMachineBlack) and (PositionFrame.Position.Active = ActiveWhite));{или – белыми }

RunThinker := (Mode = mdTwoMachine)// или –комп играет сам с собой

or ((Mode = mdMachineWhite) and (PositionFrame.Position.Active = ActiveWhite))// соответственно, //симулируя игру белыи против себя

or ((Mode = mdMachineBlack) and (PositionFrame.Position.Active = ActiveBlack));// или черными против //себя

if not RunThinker then Exit;{флаговаая переменая не сработала –выходим }

ThreadHandle := BeginThread(nil, 8\*4096, @Thinker, @PositionFrame.Position, CREATE\_SUSPENDED, ThreadId);{Запускаем поток, а в потоке – ф-циии необходимые для старта партиии –thinker и positionframe }

SetThreadPriority(ThreadHandle, THREAD\_PRIORITY\_BELOW\_NORMAL);// приоритетность потока

ResumeThread(ThreadHandle);// остановить поток по завершении

end;

{прогнозирование ходов }

procedure TMainForm.DoDebug(var Message: TMessage);

var

Position: PPosition;// получаем позицию

begin

if Message.WPAram = 0 then Exit; // если ход возмлжен,

Position := Pointer(Message.WPAram);// получить указатель на позицию, в которую возможен ход

end;

{проверка отрисовки ходов, методом animate}

procedure TMainForm.IsAnimation(var Message: TMessage);

begin

if PositionFrame.Animate {если возможн отрисовать позиццию на доске, вернуть 1 , мначе 0(фигурирует в фунуциях до того, результат данной функции используется как флаг) }

then Message.Result := 1

else Message.Result := 0

end;

const

MAX\_LEN = 60; // Ходов на партию

procedure TMainForm.StopThinking;

{процедура выводда ИИ из стадии раздумья (результат этой процедуры используется как флаг )}

begin

TerminateThread(ThreadHandle, 0); {Уничтожаем поток, использовавшийся для мыщления ии }

CloseHandle(ThreadHandle);{закрываем поток, чтоб сохранить данные}

ThreadHandle := 0;{сбрасываем идентификатор потока}

end;

procedure TMainForm.FormDestroy(Sender: TObject);

begin

FreeAndNil(FGameHistory);// обнуляем историю игры, при закрытии

end;

procedure TMainForm.UndoMoveItemClick(Sender: TObject);

begin

{отмена хода не описана }

end;

{ TGameHistory }

procedure TGameHistory.AddWhiteMove(const Move: string);// для белых

var

NewItem: TListItem; // создаем объект ход в списке

begin

NewItem := PartyView.Items.Add; //добавелние в список ходов хода, сделанного в партиии, в список ходов

NewItem.Caption := IntToStr((MoveNo div 2) + 1);{ номер нотационнай запись хода 1 : e2}

NewItem.Subitems.Add(Move);// сама запись хода вида e3, получаем из ранее описанных процедур

PartyView.Selected := NewItem;// добавление нового хода в список

PartyView.Selected.MakeVisible(False);// прокрутка ходов

end;

procedure TGameHistory.AddBlackMove(const Move: string);// для черных

var

Item: TListItem; //получаем элемент списка

begin

Assert(MainForm.PartyView.Items.Count > 0);// считаем сделанные ходы

Item := PartyView.Items[PartyView.Items.Count-1]; так как черные всегда вторыми ходят, вычитаем 1

Item.Subitems.Add(Move);// и на позицию с полученным номером добавляем ход черных

PartyView.Selected := Item;// создаем элемнент списка(контейнер для хода), добавляем ход

PartyView.Selected.MakeVisible(False);// прокручиваем список

end;

procedure TGameHistory.AddMove(NewPosition: TPosition);

{запоминаем позицию }

var

Move: string; //строка- позиция

begin

Move := GetLastMove(NewPosition);{в нее записываем полученную позицию}

if Move <> '' then// если позиция не пуста

if FPositions[MoveNo].Active = ActiveWhite// проверяем, чей зод

then AddWhiteMove(Move)// если белых, то делаем ход и добавляем в список

else AddBlackMove(Move);.{иначе – то же самое за череых}

MoveNo := MoveNo + 1;// в их случае – добавляем 1 к номерации ходов

FPositions[MoveNo] := NewPosition;// впиываем ход в историю

end;

{новая история игры }

procedure TGameHistory.NewGame;

begin

MoveNo := 0;// обнуляем номер хода

PartyView.Items.Clear;// очищаем список ходов

FPositions[0] := StartBoard; // врисываем первый сделанный ход , в историю

PositionFrame.SetPosition(StartBoard); // отрисовываем сдеданный ход

end;

function TGameHistory.GetPartyView: TListView;

begin

Result := MainForm.PartyView;// фиксация партии(ходы), использовано выше

end;

function TGameHistory.GetPositionFrame: TPositionFrame;

begin

Result := MainForm.PositionFrame;// отрисовка изменений на доске

end;

procedure TGameHistory.Undo;// отмена хода

var

Last: Integer;

Item: TListItem;

begin

Assert(MoveNo > 0);// если сделан хотя бы 1 ход

MainForm.ViewItem.Click; {делегируем соотв. обработчик}

MoveNo := MoveNo - 1;// сбавляем номер хода в списке ( запись вида 3-2)

PositionFrame.SetPosition(FPositions[MoveNo], False);{меняем позицию фигур на ход нгназад оот текущей}

Last := PartyView.Items.Count-1;{меняем количество ходов}

Assert(Last >= 0);{пока оно не приблизиится к 0 }

Item := PartyView.Items[Last];

if Item.SubItems.Count > 1//(если ходов больше 1 )

then Item.SubItems.Delete(1){удаляем сделанные по 1 }

else PartyView.Items.Delete(Last); //иначе просто удадяем последний

end;

procedure TMainForm.NewGameActionExecute(Sender: TObject);

begin

StopThinking;{выводим ИИ из состояния раздумья}

GameHistory.NewGame;{инициализируем новую историю игры}

if Mode in [mdMachineWhite, mdTwoMachine] then MachineBlackItem.Click;{если режим комп белыми,, или – комп против самого себя, делаем черную часть фигур, кликабельными }

PositionFrame.AcceptMove := True;{выбираем разрешенный ход }

end;

procedure TMainForm.Deselect(Action: TAction; const Category: string);

{смена режима игры}

var

I: Integer;

begin

for I := 0 to ActionList.ActionCount - 1 do // перебрать доступные режимы

begin

if ActionList.Actions[I].Category <> Category then Continue;{реализация переключения по категориям(список режимов ) }

if ActionList.Actions[I] = Action then Continue;{выбор необходимого обработчика для категории }

(ActionList.Actions[I] as TAction).Checked := False;{сбросить отметку с тех, что не выбраны}

end;

end;

procedure TMainForm.LevelActionExecute(Sender: TObject);

begin

{переключение сложности игры по уровням}

Deselect(Sender as TAction, 'Level'); {перребор доступных уровней}

with Sender as TAction do

begin

Checked := True;

Deep := Tag;{отметка необходимого }

end;

end;

procedure TMainForm.UndoMoveActionExecute(Sender: TObject);

begin

GameHistory.Undo; {отмена хода }

end;

procedure TMainForm.ActionListUpdate(Action: TBasicAction;

var Handled: Boolean);

begin

UndoMoveAction.Enabled := GameHistory.MoveNo > 0;{контроль количества ходов в партии при отмене хода не должно уходить за 0 }

end;

procedure TMainForm.ExitActionExecute(Sender: TObject);

{выход из программы }

begin

ViewItem.Click;

Close;

end;

procedure TMainForm.MachineWhiteActionExecute(Sender: TObject);

{если окмп играет белыми,}

begin

Deselect(Sender as TAction, 'Mode');{проверка сооотв. режимов}

(Sender as TAction).Checked := True; {выбор именно нужного, при котором комп играет белыми}

if Mode = mdMachineWhite then Exit;// если не выбрали выходим

FMode := mdMachineWhite;{иначе – ставим приндительно именно его }

PositionFrame.FlipBoard := True;{перреворачиваем доску }

TuneState;{}

end;

procedure TMainForm.MachineBlackActionExecute(Sender: TObject);

begin

Deselect(Sender as TAction, 'Mode');

(Sender as TAction).Checked := True;

if Mode = mdMachineBlack then Exit;

FMode := mdMachineBlack;

PositionFrame.FlipBoard := False;

TuneState;// ищем ход, сообразный режиму

end;

procedure TMainForm.TwoMachineActionExecute(Sender: TObject);

begin

Deselect(Sender as TAction, 'Mode');{переебор режимов }

(Sender as TAction).Checked := True;{выбор нужного }

if Mode = mdTwoMachine then Exit;{не выбррали –завершаем перебор }

FMode := mdTwoMachine;{и ставим режим принудительно}

TuneState;{ищем ход, сообразно выбранным настройкам}

end;

procedure TMainForm.ViewGameActionExecute(Sender: TObject);

begin

Deselect(Sender as TAction, 'Mode');{режим 2 игрока, работает также – перебираем все}

(Sender as TAction).Checked := True;{находим нужный }

if Mode = mdView then Exit;{не нашли – прерываем перебор }

FMode := mdView;{ставим принудительно искомый режим }

ViewItem.Checked := True;{передвигаем отметку(точку)}

if ThreadHandle <> 0 then StopThinking;{запускаем мышление ИИ}

end;

procedure TMainForm.FlipBoardActionExecute(Sender: TObject);

begin

PositionFrame.FlipBoard := not PositionFrame.FlipBoard;{переворачиваем доску }

end;

procedure TMainForm.CopyGameActionExecute(Sender: TObject);

var

MoveNo: Integer;

Item: TListItem;

CurrentSt: string;

AllParty: TStringList;

{копирование партии }

procedure Add(const St: string);

begin

if Length(CurrentSt) + Length(St) + 1 > MAX\_LEN then{если партия длиннее 60 ходов, то сохранить ее в переменнную CurrentSt)}

begin

AllParty.Add(CurrentSt); {записать в список для всей партии (динамический,)}

CurrentSt := '';{строку обнулить }

end;

if CurrentSt <> '' then CurrentSt := CurrentSt + ' ';{при необходимости – перезаписать (пробелами )}

CurrentSt := CurrentSt + St;

end;

var

I, J : Integer;

F : TextFile;

S : String;

begin

AssignFile(F, 'Сохраненная партия.txt');{создать текстовой файл с партией}

Rewrite(F);{открыть его на запись }

for I := 0 to PartyView.Items.Count - 1 do{прочитать списоко ходлов }

begin

S := PartyView.Items.Item[I].Caption + ',';{записать в файл только сами ходы}

for J := 0 to PartyView.Items.Item[I].SubItems.Count - 1 do{пройти по списку и получить нумерацию ходов}

S := S + PartyView.Items.Item[I].SubItems.Strings[J] + ',';{записать в соседней с ходами строке }

SetLength(S, Length(S) - 1); {совмещаем первый ход и 1 байт строки }

WriteLn(F, S);{записать строку с номерами в файд}

end;

CloseFile(F);{закрыть файл}

end;

end.