|  |
| --- |
| unit UCreateNewAccount;  interface  uses  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Data.DB, Data.Win.ADODB,Utilities;  type  TFmCreateNewAccount = class(TForm)  LblUsername: TLabel;  LblPassword: TLabel;  LblConfirmPassword: TLabel;  EdUsername: TEdit;  EdPassword: TEdit;  EdConfirmPassword: TEdit;  BtnCreateAccount: TButton;  ADOPlayer: TADOTable;  LblTitle: TLabel;  GpBxCreate: TGroupBox;  CheckBoxExpert: TCheckBox;  procedure BtnCreateAccountClick(Sender: TObject);  procedure FormActivate(Sender: TObject);  procedure FormClose(Sender: TObject; var Action: TCloseAction);  private  { Private declarations }  public  { Public declarations }  end;  var  FmCreateNewAccount: TFmCreateNewAccount;  implementation  {$R \*.dfm}  uses ULogin;  procedure TFmCreateNewAccount.BtnCreateAccountClick(Sender: TObject);  var next:integer;  FreeSpaceFound:boolean;  begin  if AdoPlayer.Locate('Username',EdUsername.Text,[]) then Showmessage('This username has already been taken')  else if (length(EdUsername.Text)<5) or (length(EdUsername.Text)>40) then Showmessage('Username must be between 5 and 40 characters long.')  else if (length(EdPassword.Text)<5) or (length(EdPassword.Text)>25) then Showmessage('Password must be between 5 and 25 characters long.')  else if (EDPassword.Text<>EDConfirmPassword.Text) then ShowMessage('Passwords do not match.')  else if (AdoPlayer.RecordCount>=20) then ShowMessage('Database is full, contact administrator.')  else  begin  next:=1;  FreeSpaceFound:=False;  AdoPlayer.Edit;  while Not(FreeSpaceFound) do  begin  if Not(ADOPlayer.Locate('AccountID',next,[])) then FreeSpaceFound:=True  else next:=next+1;  end;  AdoPlayer.Append;  AdoPlayer['AccountID']:=next;  AdoPlayer['Username']:=EdUsername.text;  AdoPlayer['Passcode']:=EdPassword.text;  AdoPlayer['Lastlogin']:=Date();  AdoPlayer['Expert']:=CheckBoxExpert.Checked;  AdoPlayer['Wins']:=0;  AdoPlayer['Losses']:=0;  AdoPlayer.Post;  EdUsername.Text:='';  EdPassword.Text:='';  EdConfirmPassword.Text:='';  Showmessage('Saved new account.');  FmCreateNewAccount.Hide;  FmLogin.Show;  //ADOPlayers: Dataset not in edit or insert mode  end  end;  procedure TFmCreateNewAccount.FormActivate(Sender: TObject);  begin  AdoPlayer.ConnectionString:= ConnStr;  AdoPlayer.TableName:='Player';  AdoPlayer.Open;  end;  procedure TFmCreateNewAccount.FormClose(Sender: TObject;  var Action: TCloseAction);  begin  FmCreateNewAccount.Hide;  FmLogin.Show;  AdoPlayer.Close;  end;  end. |
| unit UCreateNewGame;  interface  uses  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Data.DB, Vcl.DBCtrls,  Data.Win.ADODB, Utilities;  type  TFmCreateNewGame = class(TForm)  LblCreateNewGame: TLabel;  LblFactionType: TLabel;  LblFactionCapital: TLabel;  LblFactionTrait: TLabel;  LblMapSize: TLabel;  LblMapName: TLabel;  LblSaveFileName: TLabel;  EdSaveFileName: TEdit;  GbxFaction: TGroupBox;  LblFactionTypeOutput: TLabel;  LblFactionCapitalOutput: TLabel;  LblFactionTraitOutput: TLabel;  LblFactionName: TLabel;  GbxMap: TGroupBox;  LblMapSizeOutput: TLabel;  LblSaveID: TLabel;  BtnCreateGame: TButton;  BtnCancel: TButton;  ADOMap: TADOTable;  DBCmbMap: TDBLookupComboBox;  DSMap: TDataSource;  ADOFaction: TADOTable;  DSFaction: TDataSource;  DbCmbFaction: TDBLookupComboBox;  ADOQuery1: TADOQuery;  ADOSaveState: TADOTable;  LblFactionID: TLabel;  LblFactionIdOutput: TLabel;  LblMapID: TLabel;  LblMapIDOutput: TLabel;  LblMapNumberOfFactionsOutput: TLabel;  LblMapNumberOfFactions: TLabel;  LblSaveIDOutput: TLabel;  procedure BtnCancelClick(Sender: TObject);  procedure FormActivate(Sender: TObject);  procedure FormClose(Sender: TObject; var Action: TCloseAction);  procedure DbCmbFactionClick(Sender: TObject);  procedure BtnCreateGameClick(Sender: TObject);  procedure DBCmbMapClick(Sender: TObject);  function AutoSaveID:boolean;  procedure FormShow(Sender: TObject);  private  { Private declarations }  public  { Public declarations }  end;  var  FmCreateNewGame: TFmCreateNewGame;  implementation  {$R \*.dfm}  uses UMenu, ULogin, UCurrentGame;  procedure TFmCreateNewGame.BtnCancelClick(Sender: TObject);  begin  FmCreateNewGame.Hide;  FmMenu.Show;  end;  procedure TFmCreateNewGame.BtnCreateGameClick(Sender: TObject);  begin  if (DBCmbFaction.Text='') then ShowMessage('Please enter a faction.')  else if (DBCmbMap.Text='') then ShowMessage('Please enter a map.')  else if (EdSaveFileName.Text='') then ShowMessage('Please enter a file name.')  else if (AutoSaveID) then FmCreateNewGame.Close  else  begin  AdoSaveState.Append;  if (AdoSaveState.RecordCount>60) then  ShowMessage('Database full, contact administrator.')  else  AdoSaveState['SaveID']:=LblSaveIDOutput.Caption;  AdoSaveState['FileName']:=EdSaveFileName.text;  AdoSaveState['AccountID']:=AccountID;  AdoSaveState['MapID']:=LblMapIDOutput.Caption;  AdoSaveState['FactionID']:=LblFactionIDOutput.Caption;  AdoSaveState['NumberOfTurns']:=1;  AdoSaveState['CurrentNumberOfFactions']:=LblMapNumberOfFactionsOutput.Caption;  AdoSaveState.Post;  Showmessage('Game created.');  FmCurrentGame.tag:=StrToInt(LblMapIdOutput.Caption);  FmCurrentGame.LblSaveID.Caption:=LblSaveIdOutput.Caption;  FmCurrentGame.LblMapTitle.Caption:=DBCmbMap.Text;  FmCurrentGame.LblFactionTitle.Caption:=DBCmbFaction.Text;  FmCurrentGame.LblNumberOfTurns.Caption:='1';  FmCreateNewGame.Hide;  isLoad:=False;  FmCurrentGame.Show;  end  end;  procedure TFmCreateNewGame.DbCmbFactionClick(Sender: TObject);  begin  AdoQuery1.ConnectionString:=Connstr;  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select FactionID,Type,CapitalName FROM Faction WHERE Name = :FactionName');  ADOQuery1.Parameters.ParamByName('FactionName').Value := DBCmbFaction.Text;  AdoQuery1.Open;  LblFactionIDOutput.Caption := ADOQuery1.Fields[0].AsString;  LblFactionTypeOutput.Caption := ADOQuery1.Fields[1].AsString;  LblFactionCapitalOutput.Caption := ADOQuery1.Fields[2].AsString;  end;  procedure TFmCreateNewGame.DBCmbMapClick(Sender: TObject);  begin  AdoQuery1.ConnectionString:=Connstr;  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select MapID,XCoordinate,YCoordinate,NumberOfFactions FROM Map WHERE Name = :MapName');  ADOQuery1.Parameters.ParamByName('MapName').Value := DBCmbMap.Text;  AdoQuery1.Open;  LblMapIDOutput.Caption := ADOQuery1.Fields[0].AsString;  LblMapSizeOutput.Caption := ADOQuery1.Fields[1].AsString+'x'+AdoQuery1.Fields[2].AsString;  LblMapNumberOfFactionsOutput.Caption := ADOQuery1.Fields[3].AsString;  end;  procedure TFmCreateNewGame.FormActivate(Sender: TObject);  begin  AdoMap.ConnectionString:= ConnStr;  AdoMap.TableName:='Map';  DSMap.DataSet:=AdoMap;  DBCmbMap.ListSource:=DSMap;  DBCmbMap.ListField:='Name';  DBCmbMap.KeyField:='MapID';  AdoMap.Open;  AdoFaction.ConnectionString:= ConnStr;  AdoFaction.TableName:='Faction';  DSFaction.DataSet:=AdoFaction;  DBCmbFaction.ListSource:=DSFaction;  DBCmbFaction.ListField:='Name';  DBCmbFaction.KeyField:='FactionID';  AdoFaction.Open;  AdoSaveState.ConnectionString:= ConnStr;  AdoSaveState.TableName:='SaveState';  AdoSaveState.Open;  end;  procedure TFmCreateNewGame.FormClose(Sender: TObject; var Action: TCloseAction);  begin  AdoMap.Close;  AdoFaction.Close;  AdoSaveState.Close;  FmMenu.Show;  end;  function TFmCreateNewGame.AutoSaveID:boolean;  var Full:Boolean;  begin  Full:=False;  Activate;  adosavestate.Close; adosavestate.Open;  if (ADOSaveState.Locate('SaveID;AccountID',VarArrayOf([1, AccountID]),[])) then  begin  if (ADOSaveState.Locate('SaveID;AccountID',VarArrayOf([2, AccountID]),[]))  then begin  if (ADOSaveState.Locate('SaveID;AccountID',VarArrayOf([3, AccountID]),[]))  then begin  Showmessage('You have reached the maximum number of saves. Please delete a save.');  Full:=True;  LblSaveIdOutput.Caption:='Save files full';  end  else LblSaveIdOutput.Caption:='3';  end  else LblSaveIdOutput.Caption:='2';  end  else LblSaveIdOutput.Caption:='1';  Result:=Full;  end;  procedure TFmCreateNewGame.FormShow(Sender: TObject);  begin  Activate;  AutoSaveID;  end;  end. |
| unit UCurrentGame;  interface  uses  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls, Data.Win.ADODB,comobj, Data.DB, strutils,  Vcl.Imaging.pngimage, Vcl.MPlayer, printers;  type  TFmCurrentGame = class(TForm)  TileX1Y1: TImage;  TileX2Y1: TImage;  TileX3Y1: TImage;  TileX4Y1: TImage;  TileX5Y1: TImage;  TileX1Y2: TImage;  TileX1Y3: TImage;  TileX1Y4: TImage;  TileX6Y1: TImage;  TileX7Y1: TImage;  TileX2Y2: TImage;  TileX3Y2: TImage;  TileX4Y2: TImage;  TileX5Y2: TImage;  TileX6Y2: TImage;  TileX7Y2: TImage;  TileX2Y3: TImage;  TileX3Y3: TImage;  TileX3Y4: TImage;  TileX4Y4: TImage;  TileX5Y4: TImage;  TileX6Y4: TImage;  TileX7Y4: TImage;  TileX7Y3: TImage;  TileX4Y3: TImage;  TileX6Y3: TImage;  TileX5Y3: TImage;  LblMapTitle: TLabel;  TileX2Y4: TImage;  ADOTile: TADOTable;  ADOQuery1: TADOQuery;  ADOCommand1: TADOCommand;  GbxGeneral: TGroupBox;  LblMapNameCaption: TLabel;  LblTurnCaption: TLabel;  BtnEndTurn: TButton;  LblSaveId: TLabel;  LblFactionTitle: TLabel;  GbxTile: TGroupBox;  LblFood: TLabel;  LblFoodPerTurn: TLabel;  LblGold: TLabel;  LblGoldPerTurn: TLabel;  LblHappiness: TLabel;  LblHappinessPerTurn: TLabel;  LblNameCaption: TLabel;  LblName: TLabel;  LblFoodCaption: TLabel;  LblGoldCaption: TLabel;  LblHappinessCaption: TLabel;  LblPlayingAs: TLabel;  LblNumberOfTurns: TLabel;  GbxSettlement: TGroupBox;  GbxSquadCreate: TGroupBox;  LblCurrentSettlement: TLabel;  LblCurrentSettlementCaption: TLabel;  LblNextSettlement: TLabel;  LblNextSettlementCaption: TLabel;  LblSettlementRequirementsCaption: TLabel;  LblSettlementRequirements: TLabel;  LblSettlementGoldCaption: TLabel;  BtnUpgrade: TButton;  LblSquadCaption: TLabel;  LblSquadNo: TLabel;  CmbObjective: TComboBox;  CmbTile: TComboBox;  LblSquadRequirementsCaption: TLabel;  LblSquadFood: TLabel;  LblSquadHappiness: TLabel;  LblSquadFoodCaption: TLabel;  LblSquadHappinessCaption: TLabel;  LblTileCoordinates: TLabel;  LblFoodSign: TLabel;  LblGoldSign: TLabel;  LblHappinessSign: TLabel;  LblSaveIDCaption: TLabel;  SndPlayer: TMediaPlayer;  BtnPrintTile: TButton;  procedure FormClose(Sender: TObject; var Action: TCloseAction);  procedure FormShow(Sender: TObject);  procedure FormActivate(Sender: TObject);  procedure ClearTerrain;  procedure MapTerrain(MaxXValue,MaxYValue:Integer);  procedure LoadPreviousGame(MaxXValue,MaxYValue:Integer);  procedure GenFactionList(MaxFactions:Integer; Var PlayingFactions:TArray<String>);  function CollisionCheck(CurrentTile:String):Boolean;  procedure SetTileInfo(CurrentTile, CurrentFaction:String);  procedure PlaceFactions(MaxXValue,MaxYValue:Integer);  function CheckExpert():Boolean;  function CheckGameWon():Boolean;  procedure SelectTile(Name:String);  procedure TileX1Y1Click(Sender: TObject);  procedure TileX2Y1Click(Sender: TObject);  procedure TileX3Y1Click(Sender: TObject);  procedure BtnEndTurnClick(Sender: TObject);  procedure TileX4Y1Click(Sender: TObject);  procedure BtnUpgradeClick(Sender: TObject);  procedure TileX5Y1Click(Sender: TObject);  procedure TileX6Y1Click(Sender: TObject);  procedure TileX7Y1Click(Sender: TObject);  procedure TileX1Y2Click(Sender: TObject);  procedure TileX2Y2Click(Sender: TObject);  procedure TileX3Y2Click(Sender: TObject);  procedure TileX4Y2Click(Sender: TObject);  procedure TileX5Y2Click(Sender: TObject);  procedure TileX6Y2Click(Sender: TObject);  procedure TileX7Y2Click(Sender: TObject);  procedure TileX1Y3Click(Sender: TObject);  procedure TileX2Y3Click(Sender: TObject);  procedure TileX3Y3Click(Sender: TObject);  procedure TileX4Y3Click(Sender: TObject);  procedure TileX5Y3Click(Sender: TObject);  procedure TileX6Y3Click(Sender: TObject);  procedure TileX7Y3Click(Sender: TObject);  procedure TileX1Y4Click(Sender: TObject);  procedure TileX2Y4Click(Sender: TObject);  procedure TileX3Y4Click(Sender: TObject);  procedure TileX4Y4Click(Sender: TObject);  procedure TileX5Y4Click(Sender: TObject);  procedure TileX6Y4Click(Sender: TObject);  procedure TileX7Y4Click(Sender: TObject);  procedure BtnPrintTileClick(Sender: TObject);  private  { Private declarations }  public  { Public declarations }  end;  var  FmCurrentGame: TFmCurrentGame;  implementation  {$R \*.dfm}  uses UMenu,ULoadGame,Utilities, ULogin;  procedure TFmCurrentGame.FormActivate(Sender: TObject);  begin  AdoTile.ConnectionString:= ConnStr;  AdoTile.TableName:='Tile';  AdoTile.Open;  AdoQuery1.ConnectionString:= ConnStr;  end;  procedure TFmCurrentGame.FormClose(Sender: TObject; var Action: TCloseAction);  begin  AdoTile.Close;  FmMenu.Show;  end;  procedure TFmCurrentGame.BtnPrintTileClick(Sender: TObject);  var dpmm:integer;  begin  with printer do  begin  begindoc;  dpmm:=pagewidth div 210;  with canvas do  begin  font.Name:='Calibri';  font.Size:=24;  font.Style:=[FSBold];  textout(15\*dpmm,35\*dpmm,'Triumphant!'); //Print Title  font.Name:='Arial';  font.Size:=12;  font.Style:=[];  textout(15\*dpmm,70\*dpmm,LblName.Caption+' Located at '+LblTileCoordinates.Caption); //Print Capital Information  textout(15\*dpmm,80\*dpmm,LblFoodCaption.Caption+' '+LblFood.Caption); //Print Food  textout(15\*dpmm,90\*dpmm,LblGoldCaption.Caption+' '+LblGold.Caption); //Print Food  textout(15\*dpmm,100\*dpmm,LblHappinessCaption.Caption+' '+LblHappiness.Caption); //Print Food  end;  enddoc;  end;  Showmessage('Tile '+LblTileCoordinates.Caption+' Printed.');  end;  procedure TFmCurrentGame.BtnUpgradeClick(Sender: TObject);  var NewID,NewGold,NewGoldPerTurn,NewFoodPerTurn,NewHappiness:Integer;  TileCoordinates:String;  begin  if (LblCurrentSettlement.Caption<>'City') then  begin  if (StrToInt(LblGold.Caption)>=StrToInt(LblSettlementRequirements.Caption)) then  begin  //Upgrade Settlement  NewGold:=StrToInt(LblGold.Caption)-StrToInt(LblSettlementRequirements.Caption);  TileCoordinates:=LblTileCoordinates.Caption;  //Selects information from currenttile  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select SettlementID,FoodPerTurn,GoldPerTurn,TotalHappiness FROM Tile WHERE ((TileID = :Coordinates) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('Coordinates').Value := TileCoordinates;  ADOQuery1.Parameters.ParamByName('AccID').Value := AccountID;  ADOQuery1.Parameters.ParamByName('SaveID').Value := LblSaveID.Caption;  AdoQuery1.Open;  NewID := ADOQuery1.Fields[0].AsInteger+1;  //Showmessage(IntToStr(NewID));  NewFoodPerTurn := (ADOQuery1.Fields[1].AsInteger)+(Settlement[NewID].FoodPerTurn);  //Showmessage(IntToStr(NewFoodPerTurn));  NewGoldPerTurn := (ADOQuery1.Fields[2].AsInteger)+(Settlement[NewID].GoldPerTurn);  //Showmessage(IntToStr(NewGoldPerTurn));  NewHappiness := ADOQuery1.Fields[3].AsInteger;  //Updates TotalGold so that they pay for the upgrading  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET TotalGold= :NewGold WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewGold').Value := NewGold;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := TileCoordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Updates SettlementID to next SettlementID at TileCoordinates  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET SettlementID= :NewID WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewID').Value := NewID;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := TileCoordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Updates FoodPerTurn to NewFoodPerTurn at TileCoordinates  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET FoodPerTurn= :NewFoodPerTurn WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewFoodPerTurn').Value := NewFoodPerTurn;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := TileCoordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Updates GoldPerTurn to NewGoldPerTurn at TileCoordinates  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET GoldPerTurn= :NewGoldPerTurn WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewGoldPerTurn').Value := NewGoldPerTurn;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := TileCoordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Updates \_ to New\_ at TileCoordinates  SelectTile(TileCoordinates);  Showmessage('Settlement Upgraded!');  SndPlayer.FileName:='Upgrade.mp3';  SndPlayer.Open;  SndPlayer.Play;  end  else Showmessage('You do not have enough gold.');  end  else Showmessage('You have reached the maximum settlement.');  end;  procedure TFmCurrentGame.ClearTerrain;  var CurrentYValue,CurrentXValue:Integer;  LocationName:String;  begin  //Clears map of terrain  For CurrentYValue := 1 to 4 do  begin  For CurrentXValue := 1 to 7 do  begin  LocationName:='TileX'+IntToStr(CurrentXValue)+'Y'+IntToStr(CurrentYValue);  TImage(FindComponent(LocationName)).Picture.Assign(nil);  end;  end;  end;  procedure TFmCurrentGame.MapTerrain(MaxXValue,MaxYValue:Integer);  var I,CurrentYValue,CurrentXValue,RandomNumber:Integer;  LocationName,MapName:String;  begin  //Maps terrain  LblMapTitle.Caption:=Map[tag].Name;  MapName:= Map[tag].Name+'.png';  For CurrentYValue := 1 to MaxYValue do  begin  For CurrentXValue := 1 to MaxXValue do  begin  LocationName:='TileX'+IntToStr(CurrentXValue)+'Y'+IntToStr(CurrentYValue);  if (Map[tag].Name='Chaos') then  begin  Randomize;  RandomNumber:=Random(4)+1;  if (RandomNumber=1) then MapName:='Island'  else if (RandomNumber=2) then MapName:='Desert'  else if (RandomNumber=3) then MapName:='Jungle'  else MapName:='Tundra';  end  else  begin  MapName:= Map[tag].Name;  end;  //Appends to tile table in database  AdoTile.Append;  AdoTile['TileID']:='('+IntToStr(CurrentXValue)+','+IntToStr(CurrentYValue)+')';  AdoTile['SaveID']:=StrToInt(LblSaveId.Caption);  AdoTile['AccountID']:=AccountID;  AdoTile['Name']:=MapName;  //Identifies what tile it is  if (Map[tag].Name='Chaos') then  begin  AdoTile['TotalFood']:=TileSet[RandomNumber].TotalFood;  AdoTile['FoodPerTurn']:=TileSet[RandomNumber].FoodPerTurn;  AdoTile['TotalGold']:=TileSet[RandomNumber].TotalGold;  AdoTile['GoldPerTurn']:=TileSet[RandomNumber].GoldPerTurn;  AdoTile['TotalHappiness']:=TileSet[RandomNumber].TotalHappiness;  AdoTile['HappinessPerTurn']:=TileSet[RandomNumber].HappinessPerTurn;  end;  for I := 1 to 4 do  begin  if (TileSet[I].Name=Map[tag].Name) then  begin  AdoTile['TotalFood']:=TileSet[I].TotalFood;  AdoTile['FoodPerTurn']:=TileSet[I].FoodPerTurn;  AdoTile['TotalGold']:=TileSet[I].TotalGold;  AdoTile['GoldPerTurn']:=TileSet[I].GoldPerTurn;  AdoTile['TotalHappiness']:=TileSet[I].TotalHappiness;  AdoTile['HappinessPerTurn']:=TileSet[I].HappinessPerTurn;  end;  end;  AdoTile['MapID']:=tag;  AdoTile.Post;  TImage(FindComponent(LocationName)).Picture.LoadFromFile(MapName+'.png');  //Append to database tile info  end;  end;  end;  function StrInArray(Var Word : String;Var ArrayOfString : Array of String) : Boolean;  var  Loop : String;  begin  for Loop in ArrayOfString do  begin  if Word = Loop then  begin  Exit(true);  end;  end;  result := false;  end;  procedure TFmCurrentGame.GenFactionList(MaxFactions:Integer; Var PlayingFactions:TArray<String>);  //Dynamic array due to not knowing MaxFactions until runtime  //Dynamic arrays start at 0  var I,RandomInt:Integer;  RandomFaction,ListOfFactions:String;  begin  Randomize;  //Sets length of array  SetLength(PlayingFactions, (MaxFactions));  //Sets player chosen faction at top of list  PlayingFactions[0]:=LblFactionTitle.Caption;  Showmessage('You are playing as: '+PlayingFactions[0]);  if MaxFactions=2 then  begin  Repeat  RandomInt:=Random(6);  RandomFaction:=Faction[RandomInt].Name;  Until (Not(StrInArray(RandomFaction,PlayingFactions)));  PlayingFactions[1]:=RandomFaction;  ListOfFactions:=ListOfFactions+'-'+PlayingFactions[1];  end  else  begin  for I := 1 to (MaxFactions-1) do  begin  Repeat  RandomInt:=Random(6);  RandomFaction:=Faction[RandomInt].Name;  Until (Not(StrInArray(RandomFaction,PlayingFactions)));  PlayingFactions[I]:=RandomFaction;  ListOfFactions:=ListOfFactions+'-'+PlayingFactions[I]+sLineBreak;  end;  end;  ShowMessage('You will be against: '+ListOfFactions);  end;  function TFmCurrentGame.CollisionCheck(CurrentTile:String):Boolean;  var I:Integer;  CollisionCoords:String;  YesNo:Boolean;  begin  YesNo:=False;  for I := 0 to 5 do  begin  //Finds the tile coordinates of each faction on this save on this account  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select TileID FROM Tile WHERE ((Name = :CFaction) AND ' +  '(AccountID = :AccId) AND (SaveID = :SaveId))');  ADOQuery1.Parameters.ParamByName('CFaction').Value := Faction[I].Name;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.Open;  CollisionCoords:=ADOQuery1.Fields[0].AsString;  //if (CollisionCoords<>'') then Showmessage('Potential Collision between '+CollisionCoords+' and '+CurrentTile);  if (CollisionCoords=CurrentTile) then  begin  //Showmessage('Collision Detected between '+CollisionCoords+' and '+CurrentTile);  YesNo:=True;  end;  end;  Result:=YesNo;  end;  procedure TFmCurrentGame.SetTileInfo(CurrentTile, CurrentFaction:String);  var FactionId:Integer;  begin  //Error comes from tile name not being updated to faction therefore cannot find collision  //Updates CurrentTile's Name to the CurrentFaction's name  {If adotile.locate('TileId;AccountID;SaveId',VarArrayOf([currenttile, accountid,lblsaveid.caption]),[]) then  begin  AdoTile.Edit;  AdoTile['Name']:=CurrentFaction;  //Selects the FactionID from the name of the faction in the array PlayingFactions  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select FactionID FROM Faction WHERE Name = :CurrentFaction');  ADOQuery1.Parameters.ParamByName('CurrentFaction').Value := CurrentFaction;  AdoQuery1.Open;  FactionID := ADOQuery1.Fields[0].AsInteger;  AdoTile['FactionID']:=FactionID;  AdoTile['SettlementID']:=1;  AdoTile.Post;  AdoTile.Refresh;  end; }  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET Name= :CurrentFaction WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('CurrentFaction').Value := CurrentFaction;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := CurrentTile;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Selects the FactionID from the name of the faction in the array PlayingFactions  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select FactionID FROM Faction WHERE Name = :CurrentFaction');  ADOQuery1.Parameters.ParamByName('CurrentFaction').Value := CurrentFaction;  AdoQuery1.Open;  FactionID := ADOQuery1.Fields[0].AsInteger;  //Updates CurrentTile's FactionID to the FactionID of the CurrentFaction WRONG FACTIONID  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET FactionID= :FactId WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('FactId').Value := FactionID;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := CurrentTile;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Updates CurrentTile's Settlement related information to correct information  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET SettlementID=1 WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := CurrentTile;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET FoodPerTurn=:NewFoodPerTurn WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewFoodPerTurn').Value := Settlement[1].FoodPerTurn;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := CurrentTile;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET GoldPerTurn=:NewGoldPerTurn WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewGoldPerTurn').Value := Settlement[1].GoldPerTurn;  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := CurrentTile;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  end;  //error is/was that the tile record is not being updated  procedure TFmCurrentGame.PlaceFactions(MaxXValue,MaxYValue:Integer);  var I,MaxFactions,MaxPlayingFactions,TempYValue,TempXValue:Integer;  LocationName,CurrentFaction,CurrentTile:String;  Occupied:Boolean;  PlayingFactions:TArray<String>;  begin  MaxFactions:=Map[tag].NumberOfFactions;  GenFactionList(MaxFactions,PlayingFactions);  //This is where for loop needs to change into array of playing factions (1,2,4,6 not 1,2,3,4)  MaxPlayingFactions:=Length(PlayingFactions);  for I := 0 to (MaxPlayingFactions-1) do  begin  //Validates whether the randomly selected tile is already occupied  Occupied:=True;  repeat  TempXValue:=Random(MaxXValue)+1;  TempYValue:=Random(MaxYValue)+1;  LocationName:='TileX'+IntToStr(TempXValue)+'Y'+IntToStr(TempYValue);  CurrentTile:='('+IntToStr(TempXValue)+','+IntToStr(TempYValue)+')';  //Runs through all factions in database to check for collision  Occupied:=CollisionCheck(CurrentTile);  until (Occupied=False);  CurrentFaction:=PlayingFactions[I];  SetTileInfo(CurrentTile,CurrentFaction);  //Changes picture of tile from terrain to base of faction  TImage(FindComponent(LocationName)).Picture.LoadFromFile(CurrentFaction+'.png');  end;  end;  procedure TFmCurrentGame.SelectTile(Name:string);  var FactionID, SettlementID:Integer;  begin  LblTileCoordinates.Caption:=Name;  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select Name,TotalFood,FoodPerTurn,TotalGold,GoldPerTurn,TotalHappiness,HappinessPerTurn,FactionID,SettlementID FROM Tile WHERE ((TileID = :Coordinates) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('Coordinates').Value := Name;  ADOQuery1.Parameters.ParamByName('AccID').Value := AccountID;  ADOQuery1.Parameters.ParamByName('SaveID').Value := LblSaveID.Caption;  AdoQuery1.Open;  //ShowMessage(Name);  LblName.Caption := ADOQuery1.Fields[0].AsString;  LblFood.Caption := ADOQuery1.Fields[1].AsString;  LblFoodPerTurn.Caption := ADOQuery1.Fields[2].AsString;  LblGold.Caption := ADOQuery1.Fields[3].AsString;  LblGoldPerTurn.Caption := ADOQuery1.Fields[4].AsString;  LblHappiness.Caption := ADOQuery1.Fields[5].AsString;  LblHappinessPerTurn.Caption := ADOQuery1.Fields[6].AsString;  FactionID := ADOQuery1.Fields[7].AsInteger;  SettlementID := ADOQuery1.Fields[8].AsInteger;  if (LblFactionTitle.Caption=LblName.Caption) then  begin  LblCurrentSettlement.Caption:=Settlement[SettlementID].Name;  if (SettlementID<>4) then  begin  LblNextSettlement.Caption:=Settlement[(SettlementID+1)].Name;  LblSettlementRequirements.Caption:=IntToStr(Settlement[(SettlementID+1)].GoldToConstruct);  end  else  begin  LblNextSettlement.Caption:='###';  LblSettlementRequirements.Caption:='###';  end;  GbxSettlement.Show;  GbxSquadCreate.Show;  end  else  begin  GbxSettlement.Hide;  GbxSquadCreate.Hide;  end;  end;  procedure TFmCurrentGame.TileX1Y1Click(Sender: TObject);  begin  selectTile('(1,1)');  end;  procedure TFmCurrentGame.TileX1Y2Click(Sender: TObject);  begin  selectTile('(1,2)');  end;  procedure TFmCurrentGame.TileX1Y3Click(Sender: TObject);  begin  selectTile('(1,3)');  end;  procedure TFmCurrentGame.TileX1Y4Click(Sender: TObject);  begin  selectTile('(1,4)');  end;  procedure TFmCurrentGame.TileX2Y1Click(Sender: TObject);  begin  selectTile('(2,1)');  end;  procedure TFmCurrentGame.TileX2Y2Click(Sender: TObject);  begin  selectTile('(2,2)');  end;  procedure TFmCurrentGame.TileX2Y3Click(Sender: TObject);  begin  selectTile('(2,3)');  end;  procedure TFmCurrentGame.TileX2Y4Click(Sender: TObject);  begin  selectTile('(2,4)');  end;  procedure TFmCurrentGame.TileX3Y1Click(Sender: TObject);  begin  selectTile('(3,1)');  end;  procedure TFmCurrentGame.TileX3Y2Click(Sender: TObject);  begin  selectTile('(3,2)');  end;  procedure TFmCurrentGame.TileX3Y3Click(Sender: TObject);  begin  selectTile('(3,3)');  end;  procedure TFmCurrentGame.TileX3Y4Click(Sender: TObject);  begin  selectTile('(3,4)');  end;  procedure TFmCurrentGame.TileX4Y1Click(Sender: TObject);  begin  selectTile('(4,1)');  end;  procedure TFmCurrentGame.TileX4Y2Click(Sender: TObject);  begin  selectTile('(4,2)');  end;  procedure TFmCurrentGame.TileX4Y3Click(Sender: TObject);  begin  selectTile('(4,3)');  end;  procedure TFmCurrentGame.TileX4Y4Click(Sender: TObject);  begin  selectTile('(4,4)');  end;  procedure TFmCurrentGame.TileX5Y1Click(Sender: TObject);  begin  selectTile('(5,1)');  end;  procedure TFmCurrentGame.TileX5Y2Click(Sender: TObject);  begin  selectTile('(5,2)');  end;  procedure TFmCurrentGame.TileX5Y3Click(Sender: TObject);  begin  selectTile('(5,3)');  end;  procedure TFmCurrentGame.TileX5Y4Click(Sender: TObject);  begin  selectTile('(5,4)');  end;  procedure TFmCurrentGame.TileX6Y1Click(Sender: TObject);  begin  selectTile('(6,1)');  end;  procedure TFmCurrentGame.TileX6Y2Click(Sender: TObject);  begin  selectTile('(6,2)');  end;  procedure TFmCurrentGame.TileX6Y3Click(Sender: TObject);  begin  selectTile('(6,3)');  end;  procedure TFmCurrentGame.TileX6Y4Click(Sender: TObject);  begin  selectTile('(6,4)');  end;  procedure TFmCurrentGame.TileX7Y1Click(Sender: TObject);  begin  selectTile('(7,1)');  end;  procedure TFmCurrentGame.TileX7Y2Click(Sender: TObject);  begin  selectTile('(7,2)');  end;  procedure TFmCurrentGame.TileX7Y3Click(Sender: TObject);  begin  selectTile('(7,3)');  end;  procedure TFmCurrentGame.TileX7Y4Click(Sender: TObject);  begin  selectTile('(7,4)');  end;  procedure TFmCurrentGame.BtnEndTurnClick(Sender: TObject);  var  X,MaxXValue,Y,MaxYValue: Integer;  Coordinates:String;  begin  if (StrToInt(LblNumberOfTurns.Caption)>=20) then  begin  ShowMessage('You have run out of time! '+sLineBreak+'You have been Defeated.');  FmLogin.Close;  end  else  begin  MaxXValue:=Map[tag].XCoordinate;  MaxYValue:=Map[tag].YCoordinate;  for Y := 1 to MaxYValue do  begin  for X := 1 to MaxXValue do  begin  //Refreshes Information  Coordinates:='('+IntToStr(X)+','+IntToStr(Y)+')';  SelectTile(Coordinates);  //Updates Food  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET TotalFood= :NewFood WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewFood').Value :=StrToInt(LblFood.Caption)+StrToInt(LblFoodPerTurn.Caption);  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := Coordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Updates Gold  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET TotalGold= :NewGold WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewGold').Value :=StrToInt(LblGold.Caption)+StrToInt(LblGoldPerTurn.Caption);  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := Coordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  //Updates Happiness  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE Tile SET TotalHappiness= :NewHappiness WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewHappiness').Value :=StrToInt(LblHappiness.Caption)+StrToInt(LblHappinessPerTurn.Caption);  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := Coordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  end;  end;  //Update Turns to +1  AdoQuery1.Close;  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('UPDATE SaveState SET NumberOfTurns= :NewTurns WHERE ((AccountID = :AccID) AND (SaveId = :SaveID))');  ADOQuery1.Parameters.ParamByName('NewTurns').Value :=StrToInt(LblNumberOfTurns.Caption)+1;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SaveId').Value := LblSaveId.Caption;  AdoQuery1.ExecSQL;  AdoTile.Refresh;  LblNumberOfTurns.Caption:=IntToStr(StrToInt(LblNumberOfTurns.Caption)+1);  //Empty Information  LblName.Caption:='';  LblFood.Caption:='';  LblFoodPerTurn.Caption:='';  LblGold.Caption:='';  LblGoldPerTurn.Caption:='';  LblHappiness.Caption:='';  LblHappinessPerTurn.Caption:='';  LblTileCoordinates.Caption:='( , )';  GbxSettlement.Hide;  GbxSquadCreate.Hide;  Showmessage('Saved Game. You are now on Turn '+LblNumberOfTurns.Caption+'.');  end;  end;  procedure TFmCurrentGame.LoadPreviousGame(MaxXValue,MaxYValue:Integer);  var X,Y:integer;  Coordinates,TilePicture,LocationName:String;  begin  for Y := 1 to MaxYValue do  begin  for X := 1 to MaxXValue do  begin  Coordinates:='('+IntToStr(X)+','+IntToStr(Y)+')';  //Showmessage('Coordinates: '+Coordinates);  LocationName:='TileX'+IntToStr(X)+'Y'+IntToStr(Y);  //Showmessage('LocationName: '+LocationName);  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select Name FROM Tile WHERE ((TileID = :CurrentTile) AND (AccountID = :AccID) AND (SaveId = :SavID))');  ADOQuery1.Parameters.ParamByName('CurrentTile').Value := Coordinates;  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SavId').Value := LblSaveID.Caption;  AdoQuery1.Open;  TilePicture:=(ADOQuery1.Fields[0].AsString)+'.png';  //Showmessage('TilePicture: '+TilePicture);  if (TilePicture<>'') then  //TImage(FindComponent(LocationName)).Picture.LoadFromFile(MapName+'.png');  TImage(FindComponent(LocationName)).Picture.LoadFromFile(TilePicture);  end;  end;  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select FactionID,NumberOfTurns FROM SaveState WHERE ((AccountID = :AccID) AND (SaveId = :SavID))');  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SavId').Value := LblSaveID.Caption;  AdoQuery1.Open;  LblFactionTitle.Caption:=Faction[ADOQuery1.Fields[0].AsInteger].Name;  LblNumberOfTurns.Caption:=ADOQuery1.Fields[1].AsString;  end;  function TFmCurrentGame.CheckExpert():Boolean;  var QueryResult:boolean;  begin  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select Expert FROM Player WHERE AccountID = :AccID');  ADOQuery1.Parameters.ParamByName('AccID').Value := AccountID;  AdoQuery1.Open;  QueryResult:=ADOQuery1.Fields[0].AsBoolean;  Result:=QueryResult;  end;  function TFmCurrentGame.CheckGameWon():Boolean;  var YesNo:boolean;  begin  YesNo:=True;  Result:=YesNo;  end;  procedure TFmCurrentGame.FormShow(Sender: TObject);  var MaxXValue,MaxYValue:Integer;  begin  //error: faction collision  //Error: need to make maps actually load  //Random List of Factions Chosen, -1 due to player selected faction must always be included  //Idea: When faction controls a city on a tile its recultivated into a new .png  //Hides faction related boxes  GbxSettlement.Hide;  GbxSquadCreate.Hide;  //Sets values  FmCurrentGame.Activate;  MaxXValue:=Map[tag].XCoordinate;  MaxYValue:=Map[tag].YCoordinate;  //If player is not an expert then they will be informed of rules and win condition(s)  if (Not(CheckExpert)) then  ShowMessage('How To Play: -Click on a tile to display information about that tile.'+sLineBreak  +'-When you are finished with your turn, click the end turn button.'+sLineBreak  +'How To Win: '+sLineBreak+'-Destroy all enemy faction bases.');  if Not(isLoad) then  begin  //Clears terrain, creates the map and creates the faction list and places maps tiles  ClearTerrain;  MapTerrain(MaxXValue,MaxYValue);  PlaceFactions(MaxXValue,MaxYValue);  end else  begin  //Clears terrain then loads previous information from a save state specified  ClearTerrain;  LoadPreviousGame(MaxXValue,MaxYValue);  end;  end;  end. |
| unit ULoadGame;  interface  uses  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,  Vcl.Imaging.pngimage, Data.DB, Data.Win.ADODB, utilities;  type  TFmLoadGame = class(TForm)  LblTitle: TLabel;  BtnSaveFile1: TButton;  GbxSave1: TGroupBox;  LblFileName1: TLabel;  LblFactionName1: TLabel;  LblMapName1: TLabel;  LblNumberOfTurns1: TLabel;  LblFileName1Caption: TLabel;  LblMapName1Caption: TLabel;  LblFactionName1Caption: TLabel;  LblNumberOfTurns1Caption: TLabel;  GbxSave2: TGroupBox;  LblFileName2: TLabel;  LblMapName2: TLabel;  LblNumberOfTurns2: TLabel;  LblFactionName2Caption: TLabel;  LblFileName2Caption: TLabel;  LblMapName2Caption: TLabel;  LblNumberOfTurns2Caption: TLabel;  BtnSaveFile2: TButton;  GbxSave3: TGroupBox;  LblFactionName3: TLabel;  LblFileName3: TLabel;  LblMapName3: TLabel;  LblNumberOfTurns3: TLabel;  LblFactionName3Caption: TLabel;  LblFileName3Caption: TLabel;  LblMapName3Caption: TLabel;  LblNumberOfTurns3Caption: TLabel;  BtnSaveFile3: TButton;  ImLogo: TImage;  ADOQuery1: TADOQuery;  BtnDeleteFile3: TButton;  LblFactionName2: TLabel;  BtnDeleteFile1: TButton;  BtnDeleteFile2: TButton;  ADOSaveState: TADOTable;  ADOTile: TADOTable;  procedure FormClose(Sender: TObject; var Action: TCloseAction);  procedure FormShow(Sender: TObject);  procedure BtnSaveFile1Click(Sender: TObject);  procedure BtnSaveFile2Click(Sender: TObject);  procedure BtnSaveFile3Click(Sender: TObject);  procedure LoadingGame(SaveID:Integer);  procedure DeletingGame(SaveID:Integer);  procedure BtnDeleteFile1Click(Sender: TObject);  procedure BtnDeleteFile3Click(Sender: TObject);  procedure FormActivate(Sender: TObject);  procedure BtnDeleteFile2Click(Sender: TObject);  private  { Private declarations }  public  { Public declarations }  end;  var  FmLoadGame: TFmLoadGame;  implementation  {$R \*.dfm}  uses UMenu, UCurrentGame, ULogin;  procedure TFmLoadGame.LoadingGame(SaveID:Integer);  var MapID:Integer;  begin  //select query to find MapID  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select MapID FROM SaveState WHERE ((AccountID = :AccID) AND (SaveId = :SavID))');  ADOQuery1.Parameters.ParamByName('AccId').Value := AccountId;  ADOQuery1.Parameters.ParamByName('SavId').Value := SaveID;  AdoQuery1.Open;  MapID:=ADOQuery1.Fields[0].AsInteger;  isLoad:=True;  fmcurrentgame.tag:=MapID;  FmCurrentGame.LblSaveId.Caption:=IntToStr(SaveID);  FmCurrentGame.LblMapTitle.Caption:=Map[MapID].Name;  FmCurrentGame.Show;  FmLoadGame.Hide;  end;  procedure TFmLoadGame.DeletingGame(SaveID: Integer);  var X,Y:Integer;  TileID:String;  begin  if ADOSaveState.Locate('SaveId;AccountID',VarArrayOf([SaveID,AccountID]),[]) then  begin  //Need to delete all tiles before the save state  for Y := 1 to 4 do  begin  for X := 1 to 7 do  begin  TileID:='('+IntToStr(X)+','+IntToStr(Y)+')';  if ADOTile.Locate('TileID;AccountID;SaveID',VarArrayOf([TileID,AccountID,SaveID]),[])  then ADOTile.Delete;  end;  end;  ADOTile.Refresh;  adotile.Close; adotile.open;  ADOSaveState.Delete;  ADOSaveState.Refresh;  TLabel(FindComponent('LblFileName'+IntToStr(SaveID))).Caption:='';  TLabel(FindComponent('LblMapName'+IntToStr(SaveID))).Caption:='';  TLabel(FindComponent('LblFactionName'+IntToStr(SaveID))).Caption:='';  TLabel(FindComponent('LblNumberOfTurns'+IntToStr(SaveID))).Caption:='';  Showmessage('Deleted Save File '+IntToStr(SaveID)+'.');  end  else Showmessage('You do not have a save file '+IntToStr(SaveID)+'.');  end;  procedure TFmLoadGame.BtnDeleteFile1Click(Sender: TObject);  begin  if (LblMapName1.Caption='') then Showmessage('Save file 1 is already empty.')  else if messageDlg('Are you sure you would like to delete save file 1?',mtconfirmation,[mbyes,mbno],0)=mryes then  DeletingGame(1);  end;  procedure TFmLoadGame.BtnDeleteFile2Click(Sender: TObject);  begin  if (LblMapName2.Caption='') then Showmessage('Save file 2 is already empty.')  else if messageDlg('Are you sure you would like to delete save file 2?',mtconfirmation,[mbyes,mbno],0)=mryes then  DeletingGame(2);  end;  procedure TFmLoadGame.BtnDeleteFile3Click(Sender: TObject);  begin  if (LblMapName3.Caption='') then Showmessage('Save file 3 is already empty.')  else if messageDlg('Are you sure you would like to delete save file 3?',mtconfirmation,[mbyes,mbno],0)=mryes then  DeletingGame(3);  end;  procedure TFmLoadGame.BtnSaveFile1Click(Sender: TObject);  begin  if (LblMapName1.Caption='') then  Showmessage('Cannot load an empty file.')  else LoadingGame(1);  end;  procedure TFmLoadGame.BtnSaveFile2Click(Sender: TObject);  begin  if (LblMapName2.Caption='') then  Showmessage('Cannot load an empty file.')  else LoadingGame(2);  end;  procedure TFmLoadGame.BtnSaveFile3Click(Sender: TObject);  begin  if (LblMapName3.Caption='') then  Showmessage('Cannot load an empty file.')  else LoadingGame(3);  end;  procedure TFmLoadGame.FormActivate(Sender: TObject);  begin  AdoSaveState.ConnectionString:= ConnStr;  AdoSaveState.TableName:='SaveState';  AdoSaveState.Open;  AdoTile.ConnectionString:= ConnStr;  AdoTile.TableName:='Tile';  AdoTile.Open;  AdoQuery1.ConnectionString:=Connstr;  end;  procedure TFmLoadGame.FormClose(Sender: TObject; var Action: TCloseAction);  begin  FmMenu.Show;  AdoSaveState.Close;  AdoTile.Close;  end;  procedure TFmLoadGame.FormShow(Sender: TObject);  var  I: Integer;  FileName:Array[1..3] of String;  FactionID:Array[1..3] of Integer;  FactionName:Array[1..3] of String;  MapID:Array[1..3] of Integer;  MapName:Array[1..3] of String;  NumberOfTurns:Array[1..3] of Integer;  begin  Activate;  //query to find filename, mapid, factionid and numberofturns for only the current account  for I := 1 to 3 do  begin  if ADOSaveState.Locate('AccountID;SaveId',VarArrayOf([AccountID,I]),[]) then  begin  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select FileName,MapID,FactionID,NumberOfTurns FROM SaveState WHERE' +  '(AccountID = '+IntToStr(AccountID)+') AND (SaveID = '+IntToStr(I)+')');  AdoQuery1.Open;  FileName[I]:=ADOQuery1.Fields[0].AsString;  MapID[I]:=ADOQuery1.Fields[1].AsInteger;  FactionID[I]:=AdoQuery1.Fields[2].AsInteger;  NumberOfTurns[I]:=ADOQuery1.Fields[3].AsInteger;  end;  end;  //query to find what map ID's link up to find the name of the map  for I := 1 to 3 do  begin  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select Name FROM Map WHERE' +  '(MapID = '+IntToStr(MapID[I])+')');  AdoQuery1.Open;  MapName[I]:=ADOQuery1.Fields[0].AsString;  end;  //query to find what faction ID's link up to find the name of the faction  for I := 1 to 3 do  begin  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select Name FROM Faction WHERE' +  '(FactionID = '+IntToStr(FactionID[I])+')');  AdoQuery1.Open;  FactionName[I]:=ADOQuery1.Fields[0].AsString;  end;  //Sets labels to their correct values  LblFileName1.Caption:=FileName[1];  LblFileName2.Caption:=FileName[2];  LblFileName3.Caption:=FileName[3];  LblMapName1.Caption:=MapName[1];  LblMapName2.Caption:=MapName[2];  LblMapName3.Caption:=MapName[3];  LblFactionName1.Caption:=FactionName[1];  LblFactionName2.Caption:=FactionName[2];  LblFactionName3.Caption:=FactionName[3];  LblNumberOfTurns1.Caption:=IntToStr(NumberOfTurns[1]);  LblNumberOfTurns2.Caption:=IntToStr(NumberOfTurns[2]);  LblNumberOfTurns3.Caption:=IntToStr(NumberOfTurns[3]);  end;  end. |
| unit ULogin;  interface  uses  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.pngimage, Vcl.StdCtrls,  Vcl.ExtCtrls, Data.Win.ADODB,comobj, Data.DB, Utilities;  type  TFmLogin = class(TForm)  ImLogo: TImage;  LblUsername: TLabel;  LblPassword: TLabel;  EdUsername: TEdit;  EdPassword: TEdit;  BtnCreateNewAccount: TButton;  BtnLogin: TButton;  GbxLogin: TGroupBox;  ImClose: TImage;  LblTitle: TLabel;  ADOCommand1: TADOCommand;  ADOPlayer: TADOTable;  ADOFaction: TADOTable;  ADOMap: TADOTable;  ADOQuery1: TADOQuery;  ADOSettlement: TADOTable;  procedure ImCloseClick(Sender: TObject);  procedure BtnCreateNewAccountClick(Sender: TObject);  procedure FormCreate(Sender: TObject);  procedure BtnLoginClick(Sender: TObject);  procedure FormActivate(Sender: TObject);  procedure FormClose(Sender: TObject; var Action: TCloseAction);  procedure FormShow(Sender: TObject);  procedure SetUpFactions(Var Faction:TFactionArray);  procedure SetUpMaps(Var Map:TMapArray);  procedure SetUpTiles(Var TileSet:TTileArray);  procedure SetUpSettlements(Var Settlement:TSettlementArray);  private  { Private declarations }  public  { Public declarations }  end;  var  FmLogin: TFmLogin;  implementation  {$R \*.dfm}  uses UCreateNewAccount, UMenu;  procedure TFmLogin.BtnCreateNewAccountClick(Sender: TObject);  begin  AdoPlayer.Close;  FmCreateNewAccount.Show;  FmLogin.Hide;  end;  procedure TFmLogin.BtnLoginClick(Sender: TObject);  var Password:string;  begin  if ((EdUsername.Text='') or (EdUsername.Text=' ')) then Showmessage('Please enter a username.')  else if ((EdPassword.Text='') or (EdPassword.Text=' ')) then Showmessage('Please enter a password.')  else  begin  AdoQuery1.ConnectionString:=Connstr;  AdoQuery1.Close; //assign new SQL expression  AdoQuery1.SQL.Clear;  AdoQuery1.SQL.Add('Select AccountID,Passcode FROM Player WHERE Username = :User');  ADOQuery1.Parameters.ParamByName('User').Value := EdUsername.Text;  AdoQuery1.Open;  AccountID := ADOQuery1.Fields[0].AsInteger;  Password := ADOQuery1.Fields[1].AsString;  if (Password=EdPassword.Text) then  begin  FmLogin.Hide;  FmMenu.Show;  end  else  begin  Showmessage('Incorrect username or password. Please try again.');  end;  end;  end;  procedure TFmLogin.FormActivate(Sender: TObject);  begin  AdoPlayer.ConnectionString:= ConnStr;  AdoPlayer.TableName:='Player';  AdoPlayer.Open;  AdoFaction.ConnectionString:= ConnStr;  AdoFaction.TableName:='Faction';  AdoFaction.Open;  AdoMap.ConnectionString:= ConnStr;  AdoMap.TableName:='Map';  AdoMap.Open;  AdoSettlement.ConnectionString:= ConnStr;  AdoSettlement.TableName:='Settlement';  AdoSettlement.Open;  end;  procedure TFmLogin.FormClose(Sender: TObject; var Action: TCloseAction);  begin  AdoPlayer.Close;  AdoFaction.Close;  AdoMap.Close;  end;  procedure TFmLogin.SetUpFactions(Var Faction:TFactionArray);  begin  Faction[0].Name := 'Creatures of The Depth';  Faction[0].TypeOfFaction:= 'Militant';  Faction[0].CapitalName := 'Eclipse';  Faction[1].Name := 'Vikings';  Faction[1].TypeOfFaction:= 'Militant';  Faction[1].CapitalName := 'Roskilde';  Faction[2].Name := 'Templars';  Faction[2].TypeOfFaction:= 'Religious';  Faction[2].CapitalName := 'Frostbound';  Faction[3].Name := 'The Hive Mind';  Faction[3].TypeOfFaction:= 'Religious';  Faction[3].CapitalName := 'Unity';  Faction[4].Name := 'Steampunkers';  Faction[4].TypeOfFaction:= 'Technological';  Faction[4].CapitalName := 'Utopia';  Faction[5].Name := 'Dwarves';  Faction[5].TypeOfFaction:= 'Technological';  Faction[5].CapitalName := 'Khazad-dûm';  end;  procedure TFmLogin.SetUpMaps(Var Map:TMapArray);  begin  Map[1].Name := 'Island';  Map[1].XCoordinate:= 3;  Map[1].YCoordinate := 1;  Map[1].NumberOfFactions := 2;  Map[2].Name := 'Desert';  Map[2].XCoordinate:= 3;  Map[2].YCoordinate := 3;  Map[2].NumberOfFactions := 3;  Map[3].Name := 'Jungle';  Map[3].XCoordinate:= 5;  Map[3].YCoordinate := 4;  Map[3].NumberOfFactions := 4;  Map[4].Name := 'Tundra';  Map[4].XCoordinate:= 6;  Map[4].YCoordinate := 3;  Map[4].NumberOfFactions := 5;  Map[5].Name := 'Chaos';  Map[5].XCoordinate:= 7;  Map[5].YCoordinate := 4;  Map[5].NumberOfFactions := 6;  end;  procedure TFmLogin.SetUpTiles(var TileSet:TTileArray);  begin  TileSet[1].Name:='Island';  TileSet[1].TotalFood:=10;  TileSet[1].FoodPerTurn:=3;  TileSet[1].TotalGold:=0;  TileSet[1].GoldPerTurn:=0;  TileSet[1].TotalHappiness:=100;  TileSet[1].HappinessPerTurn:=0;  TileSet[2].Name:='Desert';  TileSet[2].TotalFood:=1;  TileSet[2].FoodPerTurn:=1;  TileSet[2].TotalGold:=0;  TileSet[2].GoldPerTurn:=0;  TileSet[2].TotalHappiness:=100;  TileSet[2].HappinessPerTurn:=0;  TileSet[3].Name:='Jungle';  TileSet[3].TotalFood:=20;  TileSet[3].FoodPerTurn:=4;  TileSet[3].TotalGold:=0;  TileSet[3].GoldPerTurn:=0;  TileSet[3].TotalHappiness:=100;  TileSet[3].HappinessPerTurn:=0;  TileSet[4].Name:='Tundra';  TileSet[4].TotalFood:=5;  TileSet[4].FoodPerTurn:=2;  TileSet[4].TotalGold:=0;  TileSet[4].GoldPerTurn:=0;  TileSet[4].TotalHappiness:=100;  TileSet[4].HappinessPerTurn:=0;  end;  procedure TFmLogin.SetUpSettlements(Var Settlement:TSettlementArray);  begin  Settlement[1].Name:='Hamlet';  Settlement[1].Paradigm:='Core';  Settlement[1].GoldToConstruct:=5;  Settlement[1].FoodPerTurn:=5;  Settlement[1].GoldPerTurn:=1;  Settlement[2].Name:='Village';  Settlement[2].Paradigm:='Advanced';  Settlement[2].GoldToConstruct:=10;  Settlement[2].FoodPerTurn:=10;  Settlement[2].GoldPerTurn:=2;  Settlement[3].Name:='Town';  Settlement[3].Paradigm:='Complex';  Settlement[3].GoldToConstruct:=15;  Settlement[3].FoodPerTurn:=15;  Settlement[3].GoldPerTurn:=3;  Settlement[4].Name:='City';  Settlement[4].Paradigm:='Evolved';  Settlement[4].GoldToConstruct:=20;  Settlement[4].FoodPerTurn:=20;  Settlement[4].GoldPerTurn:=4;  end;  procedure TFmLogin.FormCreate(Sender: TObject);  var cat:OLEVariant;  I, MaxFactions, MaxMaps, MaxSettlements: Integer;  begin  SetUpFactions(Faction);  SetUpMaps(Map);  SetUpTiles(TileSet);  SetUpSettlements(Settlement);  MaxFactions:=5;  MaxMaps:=5;  MaxSettlements:=4;  cat:=CreateOleObject('ADOX.Catalog');  if not FileExists('Triumphant.accdb') then  begin  cat.create('Provider=Microsoft.ACE.OLEDB.12.0; Data Source=Triumphant.accdb;');  ADOCommand1.ConnectionString:=ConnStr;  //Player Table DDL  ADOCommand1.CommandText:='CREATE TABLE Player(AccountID INTEGER,' +  'Username VARCHAR(40),Passcode VARCHAR(25),LastLogin DATE,' +  'Expert BIT,Wins INTEGER,Losses INTEGER,' +  'PRIMARY KEY(AccountID))';  ADOCommand1.Execute;  //Faction Table DDL  ADOCommand1.CommandText:='CREATE TABLE Faction(FactionID INTEGER,' +  'Name VARCHAR(40),Type VARCHAR(13),CapitalName VARCHAR(20),' +  'PRIMARY KEY(FactionID))';  ADOCommand1.Execute;  //Trait Table DDL  ADOCommand1.CommandText:='CREATE TABLE Trait(TraitID INTEGER,' +  'Name VARCHAR(25),Description VARCHAR(40),' +  'FactionID INTEGER,' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'PRIMARY KEY(TraitID))';  ADOCommand1.Execute;  //Map Table DDL  ADOCommand1.CommandText:='CREATE TABLE Map(MapID INTEGER,' +  'Name VARCHAR(9),XCoordinate INTEGER,YCoordinate INTEGER,' +  'NumberOfFactions INTEGER,' +  'PRIMARY KEY(MapID))';  ADOCommand1.Execute;  //SaveState Table DDL  ADOCommand1.CommandText:='CREATE TABLE SaveState(SaveID INTEGER,' +  'FileName VARCHAR(25),' +  'AccountID INTEGER,MapID INTEGER,' +  'FactionID INTEGER,' +  'NumberOfTurns INTEGER,CurrentNumberOfFactions INTEGER,' +  'FOREIGN KEY(AccountID) REFERENCES Player(AccountID),' +  'FOREIGN KEY(MapID) REFERENCES Map(MapID),' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'PRIMARY KEY(SaveID,AccountID))';  ADOCommand1.Execute;  //Settlement Table DDL  ADOCommand1.CommandText:='CREATE TABLE Settlement(SettlementID INTEGER,' +  'Name VARCHAR(25),Paradigm VARCHAR(13),' +  'GoldToConstruct INTEGER,FoodPerTurn INTEGER,GoldPerTurn INTEGER,' +  'PRIMARY KEY(SettlementID))';  ADOCommand1.Execute;  //Tile Table DDL  ADOCommand1.CommandText:='CREATE TABLE Tile(TileID CHAR(5),SaveID INTEGER,AccountID INTEGER,' +  'Name VARCHAR(40),' +  'TotalFood FLOAT,FoodPerTurn FLOAT,' +  'TotalGold FLOAT,GoldPerTurn FLOAT,' +  'TotalHappiness INTEGER,HappinessPerTurn INTEGER,' +  'FactionID INTEGER,MapID INTEGER,' +  'SettlementID INTEGER,' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'FOREIGN KEY(MapID) REFERENCES Map(MapID),' +  'FOREIGN KEY(SettlementID) REFERENCES Settlement(SettlementID),' +  'FOREIGN KEY(SaveID,AccountID) REFERENCES SaveState(SaveID,AccountID),' +  'PRIMARY KEY(TileID,SaveID,AccountID))';  ADOCommand1.Execute;  //Squad Table DDL  ADOCommand1.CommandText:='CREATE TABLE Squad(SquadID INTEGER,' +  'Active BIT,Objective VARCHAR(10),TurnsToComplete INTEGER,' +  'XCoordinate INTEGER,YCoordinate INTEGER,FactionID INTEGER,' +  'TileID CHAR(5),SaveID INTEGER, AccountID INTEGER,' +  'FOREIGN KEY(TileID,SaveID,AccountID) REFERENCES Tile(TileID,SaveID,AccountID),' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'PRIMARY KEY(SquadID))';  ADOCommand1.Execute;  //Adds array of TFaction into factions table in database  //maybe use array of TFaction and a for loop  FmLogin.Activate;  for I := 0 to MaxFactions do  begin  AdoFaction.Append;  AdoFaction['FactionID']:=I;  AdoFaction['Name']:=Faction[I].Name;  AdoFaction['Type']:=Faction[I].TypeOfFaction;  AdoFaction['CapitalName']:=Faction[I].CapitalName;  AdoFaction.Post;  end;  //Adds array of TMap into maps table in database  for I := 1 to MaxMaps do  begin  AdoMap.Append;  AdoMap['MapID']:=I;  AdoMap['Name']:=Map[I].Name;  AdoMap['XCoordinate']:=Map[I].XCoordinate;  AdoMap['YCoordinate']:=Map[I].YCoordinate;  AdoMap['NumberOfFactions']:=Map[I].NumberOfFactions;  AdoMap.Post;  end;  for I := 1 to MaxSettlements do  begin  AdoSettlement.Append;  AdoSettlement['SettlementID']:=I;  AdoSettlement['Name']:=Settlement[I].Name;  AdoSettlement['Paradigm']:=Settlement[I].Paradigm;  AdoSettlement['GoldToConstruct']:=Settlement[I].GoldToConstruct;  AdoSettlement['FoodPerTurn']:=Settlement[I].FoodPerTurn;  AdoSettlement['GoldPerTurn']:=Settlement[I].GoldPerTurn;  AdoSettlement.Post;  end;  end;  end;  procedure TFmLogin.FormShow(Sender: TObject);  begin  FmLogin.Activate;  EdUsername.Text:='';  EdPassword.Text:='';  end;  procedure TFmLogin.ImCloseClick(Sender: TObject);  begin  AdoPlayer.Close;  AdoFaction.Close;  AdoMap.Close;  AdoSettlement.Close;  Close;  end;  end. |
| unit UMenu;  interface  uses  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.pngimage, Vcl.ExtCtrls,  Vcl.StdCtrls, Data.DB, Data.Win.ADODB, Utilities;  type  TFmMenu = class(TForm)  BtnCreateNewGame: TButton;  BtnLoadGame: TButton;  LblMenu: TLabel;  ImLogo: TImage;  BtnLogOut: TButton;  BtnDeleteAccount: TButton;  ADOPlayer: TADOTable;  ADOSaveState: TADOTable;  procedure BtnLogOutClick(Sender: TObject);  procedure BtnCreateNewGameClick(Sender: TObject);  procedure BtnLoadGameClick(Sender: TObject);  procedure FormClose(Sender: TObject; var Action: TCloseAction);  procedure BtnDeleteAccountClick(Sender: TObject);  procedure FormActivate(Sender: TObject);  private  { Private declarations }  public  { Public declarations }  end;  var  FmMenu: TFmMenu;  implementation  {$R \*.dfm}  uses ULogin, UCreateNewGame, ULoadGame;  procedure TFmMenu.BtnCreateNewGameClick(Sender: TObject);  begin  FmMenu.Close;  FmCreateNewGame.Show;  end;  procedure TFmMenu.BtnDeleteAccountClick(Sender: TObject);  var FileFound:Boolean;  I: Integer;  begin  //Checks to ensure choice to delete account wasn't accidental  if messageDlg('Are you sure you would like to delete your account?',mtconfirmation,[mbyes,mbno],0)=mryes then  begin  //Checks if Account is linked to any savestates  FileFound:=False;  for I := 1 to 3 do  begin  if ADOSaveState.Locate('SaveId;AccountID',VarArrayOf([I,AccountID]),[]) then FileFound:=True  end;  if FileFound then showmessage('Please delete all saves before deleting an account.')  else  begin  //Finds Account in database and deletes record  if ADOPlayer.Locate('AccountID',AccountID,[]) then AdoPlayer.Delete;  showmessage('Account deleted.');  FmLogin.Close;  end;  end;  end;  procedure TFmMenu.BtnLoadGameClick(Sender: TObject);  begin  FmMenu.Close;  FmLoadGame.Show;  end;  procedure TFmMenu.BtnLogOutClick(Sender: TObject);  begin  FmMenu.Close;  FmLogin.Show;  end;  procedure TFmMenu.FormActivate(Sender: TObject);  begin  AdoPlayer.ConnectionString:= ConnStr;  AdoPlayer.TableName:='Player';  AdoPlayer.Open;  AdoSaveState.ConnectionString:= ConnStr;  AdoSaveState.TableName:='SaveState';  AdoSaveState.Open;  end;  procedure TFmMenu.FormClose(Sender: TObject; var Action: TCloseAction);  begin  AdoPlayer.Close;  AdoSaveState.Close;  FmLogin.Show;  end;  end. |
| unit Utilities;  interface  uses sysutils;  Const  ConnStr='Provider=Microsoft.ACE.OLEDB.12.0; Data Source=Triumphant.accdb; Persist Security Info=False;';  Type  TFaction=record  Name:string[40];  TypeOfFaction:string[13];  CapitalName:string[20];  end;  TFactionArray=array[0..5] of Tfaction;  TMap=record  Name:string[20];  XCoordinate:Integer;  YCoordinate:Integer;  NumberOfFactions:Integer;  end;  TMapArray=array[1..5] of TMap;  TSettlement=record  Name:string[25];  Paradigm:string[20];  GoldToConstruct:Integer;  FoodPerTurn:Integer;  GoldPerTurn:Integer;  end;  TSettlementArray=array[1..4] of TSettlement;  //Name,TotalFood,FoodPerTurn,TotalGold,GoldPerTurn,TotalHappiness,HappinessPerTurn  TTileSet=record  Name:String[20];  TotalFood:Integer;  FoodPerTurn:Real;  TotalGold:Integer;  GoldPerTurn:Real;  TotalHappiness:Integer;  HappinessPerTurn:Real;  end;  TTileArray=array[1..4] of TTileSet;  Var  AccountID: Integer;  Faction:TFactionArray;  Map:TMapArray;  Settlement:TSettlementArray;  TileSet:TTileArray;  isLoad:Boolean;  implementation  end. |