Mr. Pixel

Sadržaj

[1 Uvod 3](#_Toc11088774)

[2 Korisnički interfejs 4](#_Toc11088775)

[2.1 Podešavanja 4](#_Toc11088776)

[2.2 Greške i otklanjanje grešaka 5](#_Toc11088777)

[3 RemObjects Pascal Script 6](#_Toc11088778)

[3.1 Tipovi podataka 6](#_Toc11088779)

[3.1.1 Dinamički nizovi 6](#_Toc11088780)

[3.1.2 Enumeracije 7](#_Toc11088781)

[3.1.3 Skupovi 7](#_Toc11088782)

[3.2 Komentari 7](#_Toc11088783)

[3.3 Kompajlerske direktive 8](#_Toc11088784)

[3.3.1 Inkludovanje fajlova 8](#_Toc11088785)

[3.3.2 Uslovna kompilacija 8](#_Toc11088786)

[3.4 Nativni potprogrami 8](#_Toc11088787)

[3.5 Naši potprogrami 12](#_Toc11088788)

[4 Instalacija i podešavanje razvojnog okruženja 15](#_Toc11088789)

[5 Lireratura 18](#_Toc11088790)

# Uvod

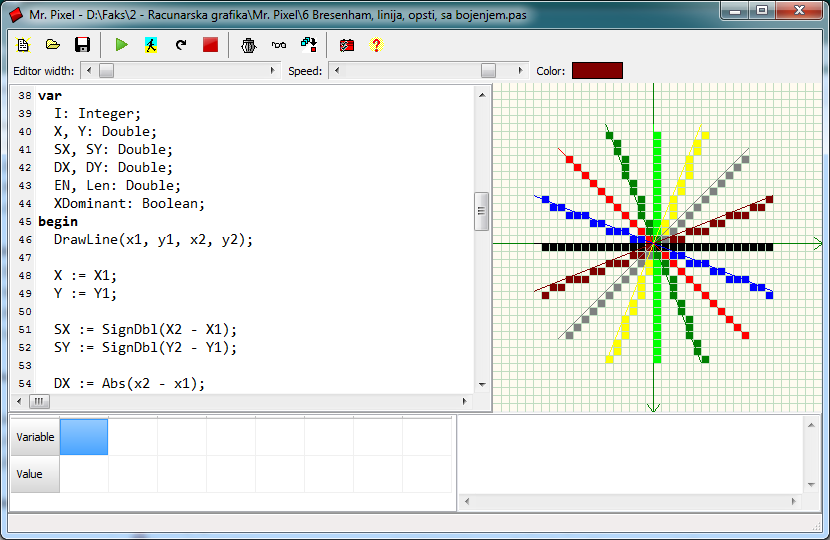
Mr. Pixel je program napisan u Delphi progamskom okruženju od strane bivšeg studenta PMF-a Branislava Stojkovića, za potrebe predmeta *Računarska grafika*. Nakon više od jedne decenije od nastanka, program se i dalje aktivno koristi na časovima računarske grafike.

Poslednje izmene nad projektom načinjene od strane autora bile su zabeležene krajem 2006. godine. Ja sam se potrudio da izbacim novu verziju sa nekim otklonjenim bagovima, poboljšanim i novim funkcionalnostima.

Prvo ću vas upoznati sa programom i načinom korišćenja, a naposletku ću dati uputstvo za instalaciju i podešavanje razvojnog okruženja.

# Korisnički interfejs

Implementacija algoritma se piše u editoru u nekoj varijanti Pascal-a. Klikom na Run (, F5) se algoritam izvršava i rezultat se iscrtava u platnu.

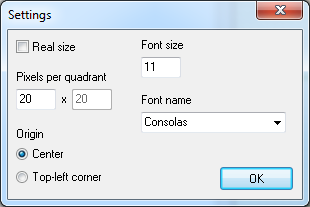


U platnu su ucrtani koordinatni sistem i mreža piksela. Pikseli su uvećani kako bi se lakše videli. Klikom na Real Size () se otvara prozorče u kome su oni prikazani u realnoj veličini.



## Podešavanja

Većina podešavanja je dostupno klikom na Settings ().

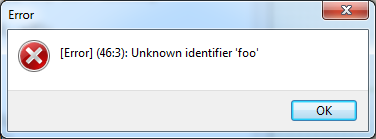


Štikliranjem kućice Real Size možete isključiti uvećan prikaz piksela. Druga dostupna podešavanja su: broj piksela po kvadrantu, lokacija koordinatnog početka, font i veličina teksta.

Ostala podešavanja su ispod trake sa alatima. Putem klizača Editor width menjate širinu editora, a brzinu bojenja piksela putem klizača Speed. Boja Color dugmeta određuje boju sledećeg piksela koji se boji. Boju menjate levim klikom na dugme (čime se otvara dijalog za izbor boja), desnim klikom na dugme (čime se bira nasumična boja), ili procedurom SetColor.

## Greške i otklanjanje grešaka

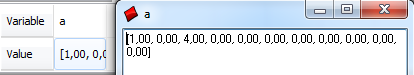
O eventualnim compile-time i runtime greškama bićete obavešteni putem iskačućeg prozorčeta gde će pisati gde u kôdu se greška javila i iz kog razloga se (otprilike) javila.



U konzoli ispod panela se prikazuju eventualna upozorenja i saveti kompajlera koje bi bilo lepo uvažiti. Svoje poruke u konzolu ispisujete procedurom WriteLn.

Izvršavanje programa je moguće pratiti korak po korak kliktanjem na Step Into (, F6) ili Step Over (, F7). Step Into je najbolje zaobići nad funkcijama koje dolaze iz inkludovanih fajlova jer se njihovo izvršavanje ne može pratiti pošto su im definicije u drugom fajlu.

U toku debagovanja u svakom koraku možete imati uvid u vrednosti svih promenljivih u opsegu. Ispod editora se nalazi tabela u kojoj u prvom redu upisujete imena promenljivih, a u drugom redu će biti prikazane njihove vrednosti. Dvoklikom na vrednost se ona prikazuje u posebnom prozorčetu. Ako kao ime promenljive upišete <stack>, prikazaće se sadržaj ugrađenog steka.



Klikom na redni broj linije (F8 dok je editor u fokusu) se postavlja breakpoint. To je tačka do koje će izvršavanje programa teći normalno, a onda će se zaustaviti i čekati na vašu dalju akciju.



# RemObjects Pascal Script

RemObjects Pascal Script je Pascal prevodilac korišćen u projektu. Podržava većinu funkcionalnosti ovog programskog jezika. Pored funkcija i procedura koje su nativno podržane od strane kompajlera, mi smo ugradili još neke funkcije i procedure koje su nam bile neophodne.

## Tipovi podataka

Najčešće korišćeni tipovi su sledeći:

|  |  |  |
| --- | --- | --- |
| Tip | Alijas | Opis |
| LongInt | Integer | -2.147.483.648 .. 2.147.483.647 |
| Double |  | 64-bitni realan broj |
| Boolean |  | logička vrednost |
| Char |  | jedan karakter |
| String | AnyString, tbtString | niz karaktera |
| Array |  | uređena kolekcija elemenata istog tipa |
| Record |  | skup više promenljivih grupisanih pod istim imenom |

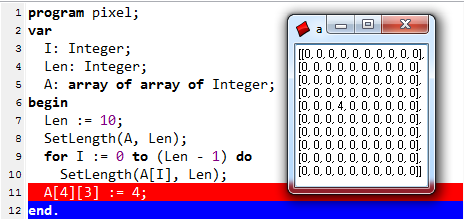
Neki slabije korišćeni tipovi su sledeći:

|  |  |  |
| --- | --- | --- |
| Tip | Alijas | Opis |
| Byte |  | 0 .. 255 |
| ShortInt |  | -128 .. 127 |
| Word |  | 0 .. 65.535 |
| SmallInt |  | -32.768 .. 32.767 |
| LongWord | Cardinal | 0 .. 4.294.967.295 |
| Int64 |  | -9.223.372.036.854.775.808 .. 9.223.372.036.854.775.807 |
| Single |  | 32-bitni realan broj |
| Extended |  | 80-bitni\* realan broj |
| Currency |  | -922.337.203.685.477,5807 .. 922.337.203.685.477,5807 |
| AnsiChar\* | Char\* | 8-bitni karakter |
| AnsiString |  | niz 8-bitnih karaktera |
| WideChar | Char\* | 16-bitni karakter |
| WideString |  | niz 16-bitnih karaktera |
| NativeString |  |  |
| UnicodeString |  |  |
| ByteBool |  | 8-bitna logička vrednost |
| WordBool |  | 16-bitna logička vrednost |
| LongBool |  | 32-bitna logička vrednost |
| Set |  | neuređena kolekcija podataka istog tipa |
| Variant |  | promenljiva koja može menjati svoj tip |

\* zavisi od računara na kome se program izvršava

### Dinamički nizovi

Dinamički nizovi su nizovi čija se veličina može menjati u toku izvršavanja programa. Deklarišu kao obični nizovi ali bez navođenja načina indeksiranja niza. Oni su uvek indeksirani su od 0, a veličina se menja putem procedure SetArrayLength.



### Enumeracije

Enumeracije ili nabrajanja su tipovi podataka definisani od strane korisnika nabrajanjem svih mogućih simbola koje promenljiva tog tipa može da uzme.

type

TMonth = (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec);

### Skupovi

Skup je neuređena kolekcija elemenata istog tipa gde se svaki element javlja najviše jednom. Pri deklarisanju skupa treba navesti sve moguće elemente. Mi smo ograničeni na skupove čiji su mogući elementi definisani enumeracijom.

Elemente u skup možemo ubaciti izjedna navođenjem između uglastih zagrada, ili ih dodavati jedan po jedan procedurom Include. Procedura Exclude se koristi za brisanje elemenata iz skupa. Za proveru da li se vrednost nalazi u skupu koristimo operator in.

program pixel;

type

TDays = (Mon, Tue, Wed, Thu, Fri, Sat, Sun);

var

I: Integer;

Weekend: Set of TDays;

Days: Array[0..3] of TDays;

begin

Weekend := [Sat, Sun];

Days[0] := Mon;

Days[1] := Sat;

Days[2] := Thu;

Days[3] := Sun;

for I := 0 to 3 do

if Days[I] in Weekend then

WriteLn('Yaaaay!')

else

WriteLn('Naaaay!')

end.

## Komentari

Komentar može početi sa // i završiti se krajem reda (jednolinijski komentari), a može biti i omeđen sa {...} ili (\*...\*) (blok komentari).

## Kompajlerske direktive

Kompajlerske direktive su naredbe koje kompajler izvršava pre nego što krene sa prevođenjem kôda. Sintaksički su to blok komentari započeti znakom $.

### Inkludovanje fajlova

Direktiva {$I fajl} ili {$INCLUDE fajl} sadržaj navedenog fajla praktično kopira na mesto na kome se ona nalazi. Korisno u slučaju ako je kôd previše veliki jer omogućava da se razbije na manje fajlove. Ja sam, recimo, implementaciju nekih korisnih matematičkih funkcija odvojio u fajl Math.pas.

Ime fajla se mora navesti između apostrofa ako sadrži razmake. Putanja do fajla je relativna u odnosu na lokaciju fajla Pixel.exe.

### Uslovna kompilacija

Upotrebom kondicionalnih direktiva govorimo prevodiocu da neki deo kôda prevodi samo pod uslovom da je vrednost datog simbola True, odnosno False. Smatra se da simbol ima vrednost False ako nije navedeno drugačije.

* {$DEFINE simbol} – postavlja vrednost simbola na True
* {$UNDEF simbol} – postavlja vrednost simbola na False
* {$IFDEF simbol} – proverava da li je vrednost simbola True
* {$IFNDEF simbol} – proverava da li je vrednost simbola False
* {$ELSE}
* {$ENDIF}

program pixel;

//{$DEFINE DARK}

var

X, Y: Integer;

begin

for Y := -5 to 5 do

for X := -5 to 5 do

begin

SetColor({$IFNDEF DARK}'clLime'{$ELSE}'clBlack'{$ENDIF});

Plot(X, Y);

end;

end.

## Nativni potprogrami

function IntToStr(I: Int64): String;

Videti ispod.

function IntToStr(I: Integer): String;

Konvertuje ceo broj u string. Neophodno pri ispisu broja u konzolu.

function StrToInt64(S: String): Int64;

Videti ispod.

function StrToInt(S: String): LongInt;

Konvertuje tekstualni zapis celog broja u ceo broj.

function StroToInt64Def(S: String, Def: Int64): Int64;

Videti ispod.

function StrToIntDef(S: String; Def: LongInt): LongInt;

Vraća konvertovan ceo broj ako je S validan tekstualni zapis celog broja, u suprotnom vraća Def.

function FloatToStr(E: Extended): String;

Konvertuje realan broj u string. Neophodno pri ispisu broja u konzolu.

function StrToFloat(S: String): Extended;

Konvertuje tekstualni zapis realnog broja u realan broj.

function Copy(S: AnyString; iFrom, iCount: LongInt): AnyString;

Kopira iCount karaktera iz stringa S počevši od pozicije iFrom. Stringovi su indeksirani od 1.

function Pos(SubStr, S: AnyString): LongInt;

Vraća indeks prvog pojavljivanja stringa SubStr u stringu S, ili 0 ako SubStr nije podstring od S. Pretraga je osetljiva na velika i mala slova.

procedure Delete(var S: AnyString; iFrom, iCount: LongInt);

Briše iCount karaktera iz stringa S počevši od pozicije iFrom.

procedure Insert(S1: AnyString; var S2: AnyString; iPos: LongInt);

Ubacuje string S1 u string S2 na poziciju iPos, šiftujući udesno sve karaktere u stringu S2 nakon te pozicije.

function StrGet(var S: String; I: Integer): Char;

Videti ispod. Ne radi na mom računaru.

function WStrGet(var S: AnyString; I: Integer): WideChar;

Vraća karakter na poziciji I u stringu S.

procedure StrSet(C: Char; I: Integer; var S: String);

Videti ispod. Ne radi na mom računaru.

procedure WStrSet(C: AnyString; I: Integer; var S: AnyString);

Menja karakter na poziciji I u stringu S u karakter C.

function AnsiUpperCase(S: String): String;

Videti ispod.

function UpperCase(S: AnyString): AnyString;

Vraća string ekvivaletan stringu S sa svim velikim slovima.

function AnsiLowerCase(S: String): String;

Videti ispod.

function LowerCase(S: AnyString): AnyString;

Vraća string ekvivaletan stringu S sa svim malim slovima.

function Trim(S: AnyString): AnyString;

Vraća string ekvivalentan stringu S sa uklonjenim svim razmacima na oba kraja stringa.

function Length(S: AnyString): Integer;

Vraća dimenzije stringa S.

procedure SetLength(var S: AnyString; NewLength: Integer);

Postavlja dimenzije stringa S na NewLength.

function PadL(S: AnyString; I: LongInt): AnyString;

Dodaje razmake sa leve strane stringa S sve dok njegova dužina ne bude jednaka I. Ne radi ništa ako string već ima tu ili veću dužinu.

function PadR(S: AnyString; I: LongInt): AnyString;

Slično kao PadL, ali razmake dodaje sa desne strane stringa.

function PadZ(S: AnyString; I: LongInt): AnyString;

Slično kao PadL, ali umesto razmaka stavlja nule.

function Replicate(C: Char; I: LongInt): String;

Videti ispod.

function StringOfChar(C: Char; I: LongInt): String;

Vraća string sačinjen od I pojavljivanja karaktera C.

procedure Dec(var X: Integer);

Umanjuje vrednost promenljive X za 1.

procedure Inc(var X: Integer);

Uvećava vrednost promenljive X za 1.

procedure SetArrayLength(Arr: Array; Count: Integer);

Postavlja dimenzije dinamičkog niza Arr na Count.

function GetArrayLength(Arr: Array): Integer;

Vraća dimenzije dinamičkog niza Arr.

function Low(Arr: Array): Integer

Vraća vrednost prvog indeksa (poziciju) u nizu.

function High(Arr: Array): Integer

Vraća vrednost poslednjeg indeksa (poziciju) u nizu.

function Ord(X: TOrdinal): LongInt;

Vraća redni broj podatka X u uređenom tipu TOrdinal. Ako je X znak, biće vraćen njegov ASCII kôd. Ako je u pitanju enumeracija, biće vraćena pozicija simbola u enumeraciji. Za cele brojeve biće vraćen sam taj broj.

function Chr(B: Byte): Char;

Inverzna funkcija funkcije Ord kada su karakteri u pitanju: vraća karakter čiji je ASCII kôd jednak B.

function Pred(X: TOrdinal): TOrdinal;

Vraća prethodinka elementa X u uređenom tipu. Izgleda da za karaktere ne radi bez dodatnih konverzija.

type

TEnum = (q, w, e, r, t, y);

var

a: Integer;

b: Char;

c: TEnum;

begin

a := Pred(10000);

b := Chr(Pred(Ord('R')));

c := Pred(r);

end.

function Succ(X: TOrdinal): TOrdinal;

Vraća sledbenika elementa X u uređenom tipu.

procedure Include(var S: Set of TSetType; M: TSetElement);

Dodaje vrednost M u skup S.

procedure Exclude(var S: Set of TSetType; M: TSetElement);

Izbacuje element M iz skupa S.

function Sin(E: Extended): Extended;

Vraća sinus ugla E u radijanima.

function Cos(E: Extended): Extended;

Vraća kosinus ugla E u radijanima.

function Sqrt(E: Extended): Extended;

Vraća koren broja E.

function Round(E: Extended): LongInt;

Vraća broj E zaokružen na najbliži ceo broj.

function Trunc(E: Extended): LongInt;

Vraća broj E sa odsečenim decimalnim mestima.

function Pi: Extended;

Vraća vrednost broja π.

function Abs(E: Extended): Extended;

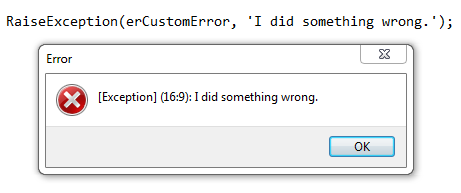
Vraća apsolutnu vrednost broja E.

procedure RaiseException(ExceptionType: TIFException; ExceptionParam: String);

Baza izuzetak tipa ExceptionType sa porukom ExceptionParam. Postoje sledeći tipovi izuzetaka:

* erNoError
* erCannotImport
* erInvalidType
* erInternalError
* erInvalidHeader
* erInvalidOpcode
* erInvalidOpcodeParameter
* erNoMainProc
* erOutOfGlobalVarsRange
* erOutOfProcRange
* erOutOfRange
* erOutOfStackRange
* erTypeMismatch
* erUnexpectedEof
* erVersionError
* erDivideByZero
* erMathError
* erCouldNotCallProc
* erOutofRecordRange
* erOutOfMemory
* erException
* erNullPointerException
* erNullVariantError
* erInterfaceNotSupported
* erCustomError

Ako se ne uhvati bačeni izuzetak dolazi do prekida izvršavanja algoritma i prikazuje se poruka u prozorčetu.



Za hvatanje izuzetaka postoje sledeće konstrukcije:

try A; except B; end;

try A; finally B; end;

try A; except B; finally C; end;

procedure RaiseLastException;

Ponovo baca poslednji bačeni izuzetak.

function ExceptionType: TIFException;

Vraća tip poslednjeg bačenog izuzetka.

function ExceptionParam: String;

Vraća poruku poslednjeg bačenog izuzetka.

function ExceptionToString(ExceptionType: TIFException; ExceptionParam: String): String;

Vraća poruku o izuzetku datog tipa.

function SizeOf(X: TAnyType): LongInt;

Vraća koliko bajtova zauzima data promenljiva.

## Naši potprogrami

function Frac(X: Double): Double;

Vraća decimalni deo realnog broja.

function RandomRange(AFrom, ATo: Integer): Integer;

Vraća nasumični broj u opsegu od AFrom do ATo, inkluzivno.

procedure ReadInt(var X: Integer);

Otvara prozor za unos celog broja.

procedure ReadFloat(var X: Double);

Otvara prozor za upis realnog broja.

function IsEmpty: Boolean;

Vraća True ako je stek prazan, inače vraća False.

procedure Push(X: Integer);

Smešta ceo broj na vrh ugrađenog steka.

procedure Pop(var X: Integer);

Uzima broj sa vrha steka.

procedure Plot(X, Y: Integer);

Boji piksel na datoj poziciji.

procedure PlotEx(X, Y, Alpha: Integer);

Boji piksel na datoj poziciji uz zadatu neprovidnost (intezitet) boje, koja može biti između 0 i 255.

procedure TurnOff(X, Y: Integer);

Isključuje piksel na datoj poziciji.

procedure Invert(X, Y: Integer);

Ako je na datoj poziciji obojen piksel, isključuje ga, a u suprotnom ga boji.

function IsOn(X, Y: Integer): Boolean;

Proverava da li je piksel na datoj poziciji obojen.

procedure DrawLine(X1, Y1, X2, Y2: Double);

Crta duž određenu tačkama *(x1, y1)* i *(x2, y2)*.

procedure DrawCircle(CX, CY: Double; R: Integer);

Crta krug poluprečnika *r* sa centrom u *(cx, cy)*.

procedure DrawEllipse(X1, Y1, X2, Y2: Double);

Crta elipsu upisanu u pravougaonik određen tačkama *(x1, y1)* i *(x2, y2)* koje se nalaze na njegovoj glavnoj dijagonali.

procedure DrawText(CX, CY: Double; Text: String);

Crta tekst na datoj poziciji.

procedure SetColorRGB(R, G, B: Byte);

Menja boju Color dugmeta. Prima vrednosti od 0 do 255.

procedure SetColor(Color: String);

Menja boju Color dugmeta. Prima HEX vrednost u $BBGGRR formatu ili naziv boje:

* clBlack
* clMaroon
* clGreen
* clOlive
* clNavy
* clPurple
* clTeal
* clGray
* clSilver
* clRed
* clLime
* clYellow
* clBlue
* clFuchsia
* clAqua
* clWhite
* clNone
* clDefault
* clScrollBar
* clBackground
* clActiveCaption
* clInactiveCaption
* clMenu
* clWindow
* clWindowFrame
* clMenuText
* clWindowText
* clCaptionText
* clActiveBorder
* clInactiveBorder
* clAppWorkSpace
* clHighlight
* clHighlightText
* clBtnFace
* clBtnShadow
* clGrayText
* clBtnText
* clInactiveCaptionText
* clBtnHighlight
* cl3DDkShadow
* cl3DLight
* clInfoText
* clInfoBk

procedure WriteLn(Text: String);

Ispisuje poruku u konzolu.

# Instalacija i podešavanje razvojnog okruženja

Projekat je inicijalno rađen u Delphi verziji 7. Uz malo izmena sam uspeo da pređem na Delphi 10.3 Rio (što je najnovija verzija u trenutku pisanja). Community izdanje Delphi-ja se može besplatno preuzeti sa zvaničnog sajta, ali je neophodno otvoriti nalog na sajtu kako bi dobili licencu.

* <https://www.embarcadero.com/products/delphi/starter/free-download>

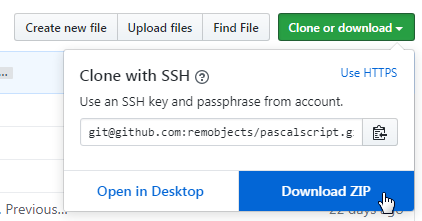
Pre otvaranja projekta morate instalirati neke korišćene nestandardne komponente i alate.

* SynEdit editor teksta (<https://github.com/SynEdit/SynEdit>)
* Pascal Script kompajler (<https://github.com/remobjects/pascalscript>)
* FastMM4 menadžer memorije (<https://github.com/pleriche/FastMM4>)

FastMM4 je alat koji prijavljuje curenja memorije pri gašenju aplikacije. Ukoliko dođe do curenja, u MrPixel direktorijumu potražite fajl Pixel\_MemoryManager\_EventLog.txt u kome će se nalaziti detaljne informacije o curenju i probajte da ga otklonite.

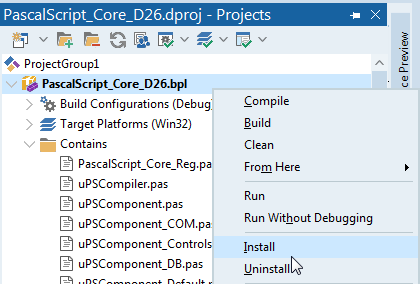
U starim verzijama projekta je korišćena jedna komponenta iz Raize Components paketa komponenti za implementaciju lupe. Iz nekog razloga lupa radi na Windows XP, a ne radi na novijim sistemima. Windows ionako dolazi uz Magnifier, te sam izbacio lupu i samim tim i dependency na Raize Components koji se plaća.

Najnovije verzije komponenti možete preuzeti na navedenim linkovima. Uz ovaj rad dolaze verzije sa kojima sam ja radio koje možete koristiti u slučaju da nove verzije nisu kompatibilne sa projektom.

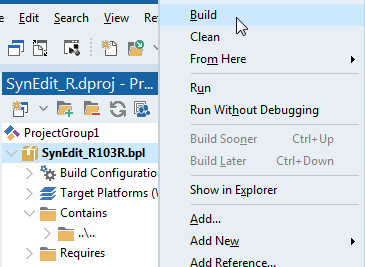


Arhive raspakujete u C:\Program Files (x86)\Embarcadero\Studio\20.0\include direktorijum (možda je na vašem računaru putanja drugačija).

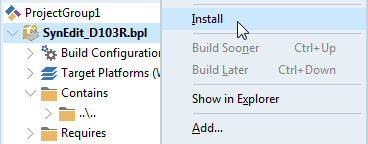
U Delphi-ju otvorite fajl PascalScript\Source\PascalScript\_Core\_D26.dproj, kliknite desnim tasterom miša na PascalScript\_Core\_D26.bpl u Projects prozoru, pa levim tasterom na Install.



Otvorite SynEdit\Packages\103R\SynEdit\_R.dproj, kliknite desnim tasterom miša na SynEdit\_R103R.bpl u Projects prozoru, pa levim tasterom na Build.

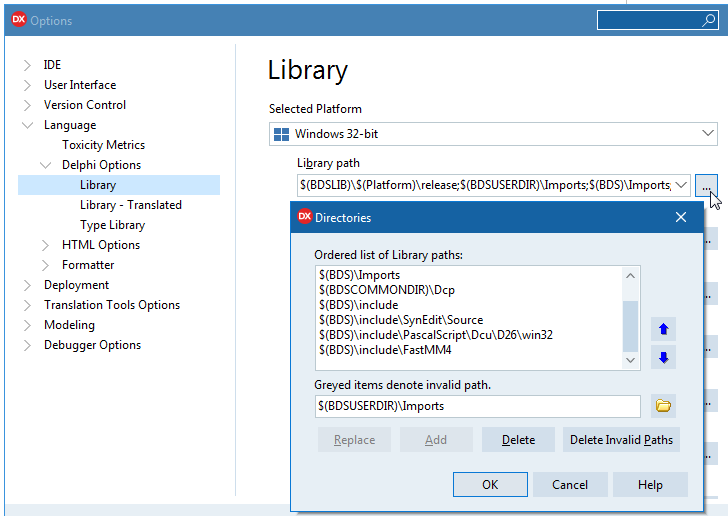


Otvorite SynEdit\Packages\103R\SynEdit\_D.dproj, kliknite desnim tasterom miša na SynEdit\_D103R.bpl u Projects prozoru, pa levim tasterom na Install.



Idite na Tools ⇨ Options... ⇨ Delphi Options ⇨ Library ⇨ Library path i dodajte sledeće putanje:

* $(BDS)\include\FastMM4
* $(BDS)\include\SynEdit\Source
* $(BDS)\include\PascalScript\Dcu\D26\win32



U tekst editoru po želji otvorite fajl FastMM4\FastMM4Options.inc. Na nekim mestima je nophodno dodati, a na nekim mestima obrisati tačku nakon otvorene vitičaste zagrade.

* {.$define FullDebugMode} obrišite tačku
* {$define RequireDebuggerPresenceForLeakReporting} dodajte tačku

Ako naknadno menjate ovaj fajl, svaki put ćete morati da izbildate sve projekte (a ne samo MrPixel) klikom na Project ⇨ Build All Projects.

Pronađite i prekopirajte fajl FastMM4\FullDebugMode DLL\Precompiled\FastMM\_FullDebugMode.dll u MrPixel direktorijum.

# Lireratura

1. Павел Дуборкин, *Справочник по RemObjects Pascal Script*  
   <http://mydataexpress.ru/remobjects_pascal_script.pdf>
2. Jitendra Kumar, *FASTMM and how to use in a Delphi Project*

<http://delphiprogrammingdiary.blogspot.com/2018/09/fastmm-and-how-to-use-in-delphi-project.html>

1. Pascal Script Wiki  
   <https://github.com/remobjects/pascalscript/wiki>
2. Help for Precision Builder Express - *Pascal Script Syntax*  
   <http://www.be-precision.com/products/precision-builder/express/webhelp/en/topics/PSSyntax.htm>
3. Help for RAD Studio Rio  
   <http://docwiki.embarcadero.com/RADStudio/Rio/en/Data_Types,_Variables,_and_Constants_Index>  
   <http://docwiki.embarcadero.com/RADStudio/Rio/en/Conditional_compilation_(Delphi)>
4. Source Code for RemObjects Pascal Script  
   <https://github.com/remobjects/pascalscript/blob/master/Source/uPSCompiler.pas>  
   <https://github.com/remobjects/pascalscript/blob/master/Source/uPSPreProcessor.pas>