

# Oefenopgave Week 15, 2IP05 (Programmeren)

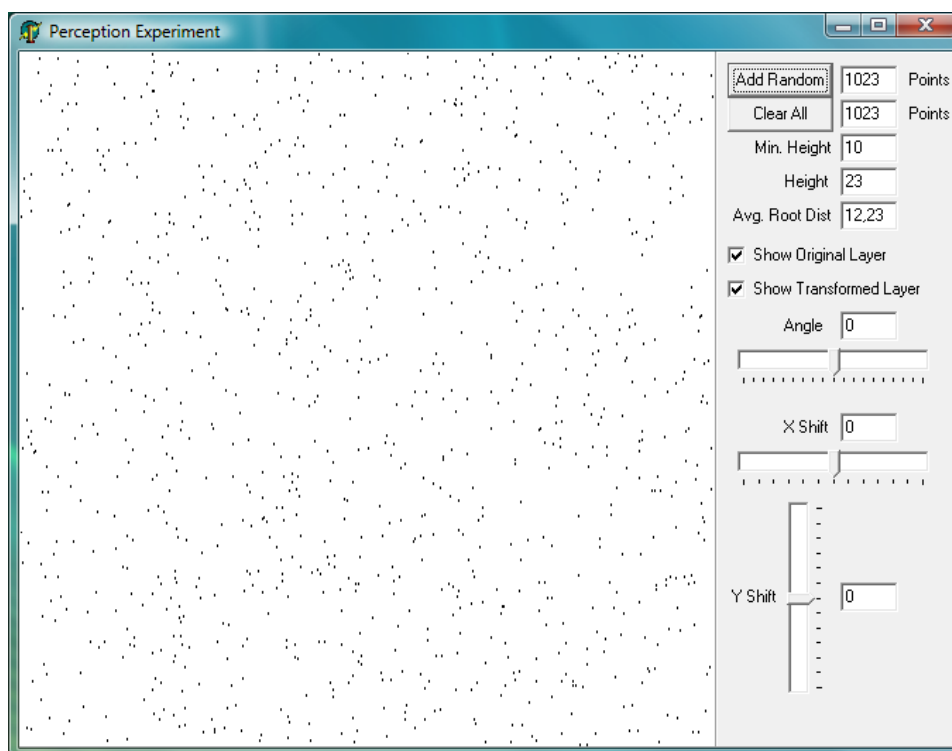
— Gebaseerd op de eindopdracht 2IP05 in 2007 —

Technische Universiteit Eindhoven  
Faculteit Wiskunde & Informatica

December 2008

## 1 Inleiding

De eindopdracht betreft het (af)maken van een applicatie voor het doen van visuele experimenten met stochastische puntverzamelingen. Daartoe heeft de applicatie een Grafisch User Interface (GUI). Tevens illustreert de applicatie enkele eigenschappen van stochastisch opgebouwde binaire zoekbomen.

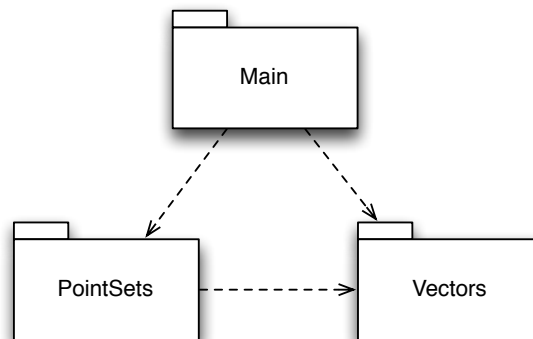


Figuur 1: 1023 willekeurige punten met de identieke transformatie

## 2 Overzicht

Bij deze opdracht hoort een archief met wat bestanden:

<code>PointsExperimentDemo.exe</code>	demo ter referentie
<code>Vectors.pas</code>	ADT voor 2D vectoren
<code>PointSets.pas</code>	ADT voor puntverzamelingen
<code>TreeViewer.pas</code>	nuttig bij zoeken van fouten
<code>TreeTester.lpr</code>	voorbeeld bij <code>TreeViewer</code>
<code>TreeViewer.out_expected</code>	verwachte uitvoer



Figuur 2: Onderlinge afhankelijkheden tussen de units

Het programma is nog niet af. Het volgende moet nog gedaan worden:

1. Creëer een nieuwe GUI applicatie: `PointsExperiment.lpi` met `Main.pas`; voeg `Vectors.pas` en `PointSets.pas` toe aan het project en aan de `uses` clause in `Main.pas`.
2. Bouw het GUI in `Main` in overeenstemming met §3 en de demo.
3. Voltooi `PointSets.pas` in overeenstemming met §4.

## 3 Grafische User Interface

Het Grafische User Interface (GUI) toont aan de linkerzijde een puntverzameling met daaroverheen, in een tweede laag, een getransformeerde versie ervan. De transformatie bestaat uit een rotatie om het midden en een verschuiving, in te stellen met de zogenaamde trackbars aan de rechterzijde. Met behulp van de checkboxes kunnen de twee lagen afzonderlijk aan- en uitgezet worden.

Met de knop `Add Random` wordt het opgegeven aantal willekeurige punten aan de puntverzameling toegevoegd. Bij opstarten staan er drie punten:

één in het midden en twee in de X- en Y-richting ernaast (deze zijn daar geplaatst via het *OnCreate event* van het formulier).

Met de knop **Clear All** worden alle punten verwijderd. Het totaal aantal punten staat naast deze knop in een *ReadOnly TEdit*.

Ten slotte staan er rechts (in *ReadOnly TEdits*) nog drie kentallen van de binaire zoekboom waarin de punten zijn opgeslagen:

Label	Waarde
Min. Height	minimale hoogte voor dit aantal punten
Height	actuele hoogte
Avg. Root Dist.	gemiddelde wortelafstand

Een binaire boom met hoogte  $h$  heeft maximaal  $2^h - 1$  knopen. Derhalve is de hoogte van een binaire boom met  $n$  knopen minimaal  $\lceil \log_2(n + 1) \rceil$ . De gemiddelde wortelafstand wordt in §4 uitgelegd.

### 3.1 Gedetailleerdere aanwijzingen

De twee lagen om de puntverzameling en diens transformatie in te tekenen zijn van het type *TPaintBox*, zeg  $500 \times 500$ . Het is zaak om van beide het *OnPaint event* te koppelen aan een routine die het betreffende plaatje opbouwt. Dit kan door de methode *Visit* van *TPointSet* met een geschikte *TAction* aan te roepen, zeg zelfgemaakte routines *DrawLayer1* en *DrawLayer2* met een parameter van type *TVector*. Deze actuele *TAction* routines kunnen gebruik maken van *TVector.Draw*. Voor het tekenen van de originele puntverzameling (in laag 1) volstaat het alle transformatieparameters 0 te kiezen. Voor het tekenen van de getransformeerde puntverzameling (in laag 2) dienen de transformatieparameters bepaald te worden uit de stand (*Position*) van de drie trackbars en de afmetingen (*Width* en *Height*) van de paintbox.

Van de checkboxes (type *TCheckBox*) dient het *OnClick event* ingevuld te worden om de *property Visible* van de overeenkomstige paintbox te veranderen.

De trackbars zijn van het type *TTrackBar*. De ligging is in te stellen met hun *property Orientation*. Hun *OnChange event* dient gekoppeld te worden aan een routine die de bijbehorende *TEdit* invult en *Invalidate* aanroept van de paintbox voor laag 2. Dit laatste zorgt ervoor dat deze laag opnieuw weergegeven wordt en zo de nieuwe toestand toont.

De knoppen **Add Random** en **Clear All** passen de puntverzameling aan en roepen ook *Invalidate* aan van beide paintboxes. Verder dienen ze de vier kentallen aan te passen.

Het is van belang dat het GUI de benodigde functionaliteit biedt. De preciese visuele opbouw is minder van belang en hoeft niet in alle details identiek te zijn aan de meegeleverde demo.

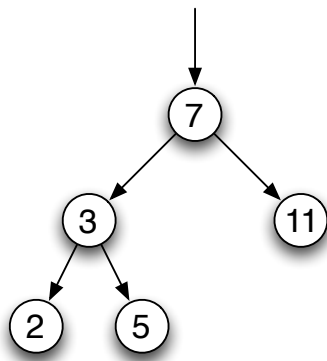
## 4 Abstract Data Type **TPointSet**

Het Abstract Data Type (ADT) **TPointSet** staat in `PointSets.pas`. Het maakt gebruik van het ADT **TVectors** uit `Vectors.pas`. Het contract van **TPointSet** is gegeven, alsmede de privé velden, privé invariant en abstractiefunctie voor de representatie. De implementatie van de methodes ontbreekt en dient tussen de markeringen

```
//# BEGIN TODO ...  
//# END TODO
```

ingevuld te worden. Laat deze markeringen zelf a.u.b. staan.

De representatie van een puntverzameling is in de vorm van een binaire zoekboom voorgesteld d.m.v. dynamische records en pointers, waarvan de definities reeds gegeven zijn. Ook hiervan dienen de operaties nog geïmplementeerd te worden.



Figuur 3: Binaire zoekboom met 5 knopen

Verder is er nog een tweetal hulpfuncties op **TPointSet** om de representatie te inspecteren (dit zijn dus geen ADT operaties). Het gaat om het bepalen van de hoogte van de zoekboom en de gemiddelde wortelafstand. De zoekboom in figuur 3 heeft hoogte 3 en de wortelafstanden zijn:

Knoop	Wortelafstand
2	2
3	1
5	2
7	0
11	1
Totaal	6

De totale wortelafstand is 6, dus de gemiddelde wortelafstand is  $6/5 = 1.2$ .

Het bepalen van de totale en gemiddelde wortelafstand is optioneel, maar niet maken kost wel 2 punten bij de beoordeling van de functionaliteit.

#### 4.1 Unit **TreeViewer**

De unit `TreeViewer` bevat een procedure `WriteTree` om een `TBinTree` (zie de unit `PointSets`) naar een tekstbestand te schrijven. Eventuele schendingen van de invarianten worden ook gemeld. Dit kan nuttig zijn bij het zoeken van fouten.

Programma `TreeTester.dpr` illustreert hoe de unit `TreeViewer` kan worden gebruikt. De verwachte uitvoer `TreeViewer.out_expected` is ook bijgevoegd. `TreeTester.dpr` kan als basis dienen voor een eigen test driver voor de unit `PointSets`.

### 5 De experimenten

Je kunt dit programma gebruiken om twee experimenten mee uit te voeren:

1. met ons visuele systeem;
2. met stochastisch opgebouwde binaire zoekbomen.

#### 5.1 Visueel systeem

1.
2.  1000
3. Zorg dat beide lagen zichtbaar zijn.
4. Stel de *Angle* in op 5 graden.
5. Wat zie je? Ons visuele systeem herkent de draaiing en bepaalt automatisch het draaicentrum. Schakel een van beide lagen tijdelijk uit en kijk wat er gebeurt. Schakel beide lagen weer aan.
6. Verschuif laag 2 naar rechts (positieve X-richting). Kun je voorspellen wat er gebeurt met het draaicentrum? Kun je het achteraf verklaren?

#### 5.2 Stochastische bomen

1.
2.  100
3. Bekijk de kentallen: vergelijk minimale met actuele hoogte en ook met de gemiddelde wortelafstand.
4. Verdubbel het aantal punten.
5. Herhaal vanaf stap 3 (denk aan copy/paste) tot je een patroon ziet.

## 6 Beoordelingscriteria

Bij de beoordeling wordt gelet op (weegfactor tussen haakjes):

- Kwaliteit van de code als programmatekst (25%)
- Kwaliteit van de functionaliteit, m.n. correctheid en efficiëntie (50%)
- Kwaliteit van de GUI (25%)

## 7 Inleveren

Controleer je werk grondig (nalezen en testen) en lever het in via peach<sup>3</sup>. Het volstaat om de volgende bestanden (namen niet veranderen) in te leveren:

- `PointsExperiment.lpi` (is er niet voor Delphi)
- `PointsExperiment.lpr` (`.dpr` voor Delphi)
- `Main.pas`
- `Main.lfm` (`.dfm` voor Delphi)
- `Main.lrs` (optioneel)
- `Vectors.pas` (ongewijzigd)
- `PointSets.pas` (deze wordt door peach<sup>3</sup> uitgebreid getest)