

Komme i gang med KPL

Av Jon Schwartz
Oversatt av Bjørn Hope og Torbjørn Skauli

Oppdatert 16. november 2005

Internett: www.kidsprogramminglanguage.com

Lenker til norske filer: www.kat.no

Komme i gang med KPL

Innhold

Innledning: Hva er programmering?	3
Hvorfor bør jeg lære å programmere med KPL?	3
Hvordan skal jeg bruke denne innføringen?.....	3
OK, vis meg et program!.....	4
Hva med datagrafikk?!.....	6
Spillgrafikk med grafiske objekter	11
Bruke variabler og løkker i KPL.....	13
Bruke KPL på datamaskinen.....	16
Neste trinn etter denne innføringen	23

Innledning: Hva er programmering?

Dataprogrammering er, kort fortalt, å **fortelle datamaskinen hva den skal gjøre**.

Dataskinner er veldig flinke til å gjøre det de blir bedt om. De gjør nøyaktig det du sier de skal gjøre. Men de har **ingen fantasi!** Så når vi skriver et program for å gi datamaskinen instruksjoner, må vi være veldig nøyaktige med å fortelle hva den skal gjøre.

Hvorfor bør jeg lære å programmere med KPL?

Ulike programmeringsspråk gir deg, programmereren, ulike måter for å fortelle hva du vil at datamaskinen skal gjøre. KPL (Kid's Programming Language), eller barnas programmeringsspråk, har flere viktige fordeler for deg som er nybegynner:

- KPL er utviklet for å gjøre det **så enkelt som mulig** å lære for en nybegynner.
- KPL er utviklet for å gjøre det **så morsomt som mulig** for deg å lære det.
- KPL er også, i motsetning til de fleste andre lærespråk, utviklet slik at det skal være så likt som mulig de språkene profesjonelle programmerere i dag bruker

Det finnes ordtak som har overlevd i mange år, og det er fordi de er sanne. Ett av dem er "**Vi må lære å gå før vi kan lære å løpe**". Programmering med KPL er å lære og gå. Når du har lært KPL, vil det være **mye** enklere å lære og løpe, uansett om du har tenkt å løpe med Java, Python, Visual Basic eller C#.

Hvordan skal jeg bruke denne innføringen?

Viktig: Dette dokumentet forutsetter forbedringer i KPL som finnes i KPL-versjoner utgitt 10. oktober 2005 eller senere. Hvis du har lastet ned KPL før 10. oktober 2005, eller hvis eksemplene ikke fungerer som beskrevet når du skriver dem inn, bør du laste ned den nyeste versjonen av KPL fra <http://www.kidsprogramminglanguage.com/download.htm>.

Hvis programmering er helt nytt for deg, og du aldri har prøvd det før, anbefaler vi at du leser denne innføringen kapittel for kapittel, og forsikrer deg om at du forstår hvert kapittel før du fortsetter til det neste. Det beste er trolig å begynne med å gå gjennom innføringen uten å bruke KPL på datamaskinen, frem til du kommer til kapitlet "**Bruke KPL på datamaskinen**". Når du studerer og lærer KPL på denne måten, kan du først lære grunnlaget uten å bli distraheret av detaljer.

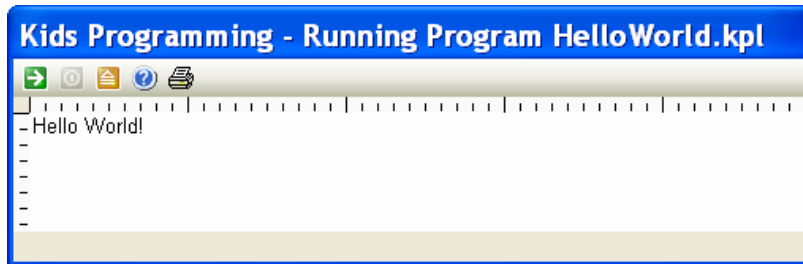
Dataprogrammering innebærer å tenke litt annerledes enn de fleste er vant til. Dataskinner krever at vi er mye mer logiske, metodiske og presise enn vi vanligvis trenger å være! Til å begynne med kan dette være litt krevende, men du vil garantert få det til! Og når du først får taket på det, blir det faktisk ganske enkelt.

En av de beste metodene for å lære å tenke på denne nye måten, er å stille spørsmål til eller få forklaringer fra noen som allerede kan dataprogrammering. Vet du om noen som kan hjelpe deg med svar og forklaringer mens du leser gjennom denne innføringen i KPL?

OK, vis meg et program!

```
1 Program HelloWorld
2   Method Main()
3     Print("Hello World!")
4   End Method
5 End Program
```

OK, det var det! Der har du det. Når du siden starter dette KPL-programmet, vil du få se følgende:



De fem linjene med KPL-programtekst du ser ovenfor er tatt fra et virkelig skjermbilde og viser hvordan programmet vises i KPL, nøyaktig slik det ser ut når du programmerer. (Et par anmerkninger fra oversetterne: 1) Programtekst kalles på engelsk "source code", og derfor brukes ofte uttrykkene "kildekode" eller bare "kode" i norsk datasjargong. 2). Alle bildene i denne oversettelsen er fra den engelske versjonen av programmet, men husk at det også finnes norske tekster til brukergrensesnittet. Disse kan lastes ned fra KPL-websidene eller www.kat.no).

Det første du kan legge merke til er linjenumrene til venstre for hver programlinje. De brukes ikke av KPL, men er der for å gi deg bedre oversikt når du arbeider med dine egne KPL-programmer. Eldre programmeringsspråk, for eksempel GWBASIC, brukte linjenumrene til å hoppe opp og ned i programmet, men KPL gjør ikke det. Et KPL-program starter på toppen og behandler én instruksjon om gangen nedover i programmet. Det finnes noen unntak fra denne regelen, men det skal vi ikke diskutere i denne innføringen.

En viktig regel når du programmerer i KPL, er at hvert ledd av programmet, hver enkelt "instruksjon" eller "utsagn", må stå på en egen linje i programmet. Eksempel: KPL-programmet under består av de samme instruksjonene som det vi nettopp så på, men siden de ikke står på hver sin linje vil ikke dette programmet fungere:

```
1 Program HelloWorld Method Main() Print("Hello World!") End Method End Program
```

Alle programmeringsspråk har bestemte regler som programmereren må følge, for at datamaskinen skal forstå programmerens instruksjoner. Av samme grunn har også kommunikasjon mellom mennesker mange regler. Vi er bare så vant til de reglene at vi følger dem uten å tenke på dem. Vi sier for eksempel ikke "Ha det" når vi tar telefonen! Og vi sier ikke "Hallo" når vi legger på. Det var kanskje et litt tullede eksempel, men det er faktisk relevant, siden KPL også krever at vi markerer begynnelsen og slutten av et program på en bestemt måte. Alle KPL-programmer må begynne med en linje tilsvarende **Program HelloWorld**, som på linje 1 i programmet under. Og alle KPL-programmer må avsluttes med **End Program**, som vist på linje 5:

```

1 Program HelloWorld
2   Method Main()
3     Print ("Hello World!")
4   End Method
5 End Program

```

Du kan kalle programmet hva du vil, men ikke bruke bokstavene æ, ø eller å i navnet. Det beste er å bruke et navn som beskriver hva programmet gjør. For dette programmet har vi valgt navnet **HelloWorld**. Vi kunne like gjerne brukt et annet navn, for eksempel **Program MyFirstProgram**.

Method Main() er en annen viktig ting du må lære om programmering i KPL. Alle KPL-programmer starter ved å kjøre den første instruksjonen etter **Method Main()**. I vårt første program er den instruksjonen **Print ("Hello World!")**.

Method Main() er en litt vilkårlig måte å definere hvor programmet skal begynne å utføre instruksjonene, men det bygger på måten de aller fleste moderne programmeringsspråk fungerer. Som du sikkert har gjettet, markerer **End Method** slutten på et avsnitt i programmet som begynte med **Method Main()**.

Vi skal ikke gå mer i detalj om metoder (**Method**) i denne innføringen, men vi håper du får et visst inntrykk av det når du ser på programeksemplet nedenfor. Det er bare én instruksjon "i" **Method Main()**, og det er linjen **Print ("Hello World!")**.

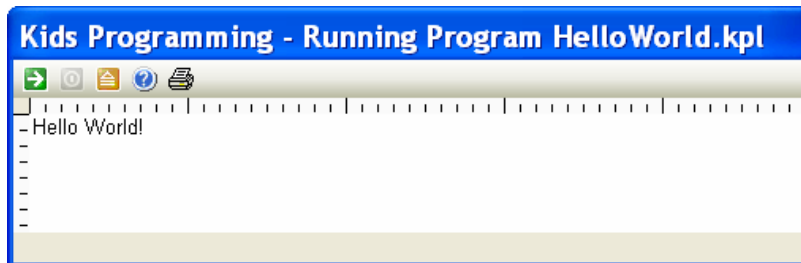
For å oppsummere, så inneholder dette programmet bare én egentlig instruksjon, nemlig **Print ("Hello World!")**. Linje 1 og 5 forteller datamaskinen hvor dette KPL-programmet begynner og slutter. Og linje 2 og 4 forteller datamaskinen hvor **Method Main()** begynner og slutter.

```

1 Program HelloWorld
2   Method Main()
3     Print ("Hello World!")
4   End Method
5 End Program

```

La oss se én gang til på vinduet som kommer til syne når du kjører dette programmet. Legg merke til at inne i dette vinduet har datamaskinen bare gjort én ting, nemlig det vi ba den gjøre med dette KPL-programmet. Datamaskinen har vist ordene **Hello World!**



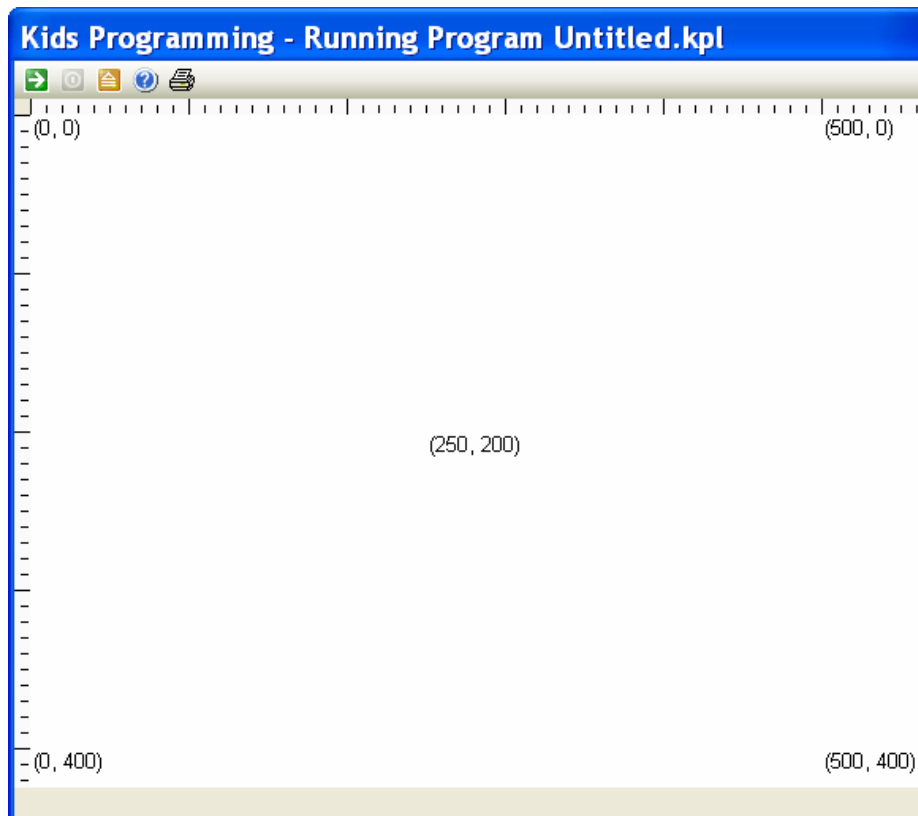
Hva med datagrafikk?!

Hello World! er et klassisk første program, men det er ikke så veldig spennende! Så la oss snakke litt om *grafikk* – det vil si å tegne på dataskjermen.

Det første vi må forklare om datagrafikk, er hvor på skjermen datamaskinen plasserer det. Datamaskiner bruker et **koordinatsystem** som er litt forskjellig fra det koordinatsystemet vi lærer om i matematikken på skolen. Men datamaskinens koordinatsystem kan være enklere å bruke, siden det gjør det enklere å arbeide med posisjoner på dataskjermen.

Datamaskiner bruker et **(X, Y)**-koordinatsystem for å angi posisjoner på dataskjermen, der **den venstre kanten av skjermen definerer $X = 0$** , og **den øvre kanten av skjermen definerer $Y = 0$** . Det betyr at **origo**, der **$X = 0$** og **$Y = 0$** , er **øvre venstre hjørne** av skjermen. Når du går mot høyre på skjermen, øker X -verdien, og når du går nedover skjermen, øker Y -verdien.

Hvis du ikke er vant til å tenke i koordinatsystem, kan denne forklaringen være litt vanskelig, så her er et bilde som viser hvor et KPL-program tegner opp en rekke **(X, Y)**-posisjoner:

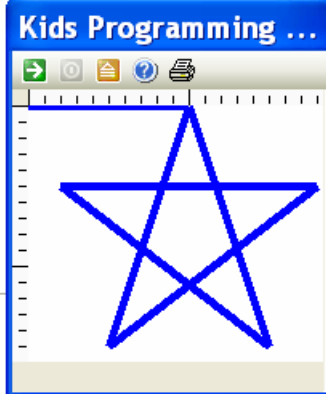


Se et øyeblikk på tallene på bildet ovenfor. Den **første verdien** i hvert tallpar er **X-verdien** for den posisjonen på skjermen. Som du kan se øker X -verdien etter hvert som du går fra venstre mot høyre. Den **andre verdien** i hvert tallpar er **Y-verdien** for den posisjonen på skjermen. Som du kan se øker Y -verdien etter hvert som du går nedover på skjermen.

Nå skal vi skrive vårt første KPL-program med grafikk, og se nøyaktig hvordan du kan bruke det grafiske koordinatsystemet til å lage grafikk på skjermen med KPL. Vi starter med å vise hele KPL-programmet,

og grafikken det lager når du kjører det. Deretter skal vi gå gjennom KPL-programmet linje for linje, for å forklare hvordan det fungerer:

```
1 Program DrawingWithThePen
2 Method Main()
3   Color(Blue)
4   PenWidth(5)
5   MoveTo(100, 0)
6   MoveTo(50, 150)
7   MoveTo(180, 50)
8   MoveTo(20, 50)
9   MoveTo(150, 150)
10  MoveTo(100, 0)
11 End Method
12 End Program
```



Legg merke til at det nå er 8 KPL-instruksjoner i **Method Main()**, fra linje 3 til linje 10. Det er mye mer enn i vårt første eksempel, men forhåpentligvis er du enig med oss i at det ikke er verst at KPL kan tegne en blå stjerne som denne på dataskjermen med bare 8 instruksjoner.

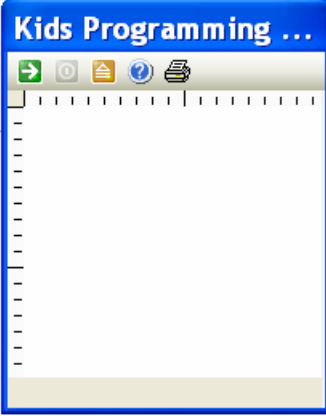
La oss se på dette programmet trinn for trinn. Legg merke til at programmet begynner og slutter på samme måte som vårt første program, bortsett fra at vi har kalt dette programmet

DrawingWithThePen:

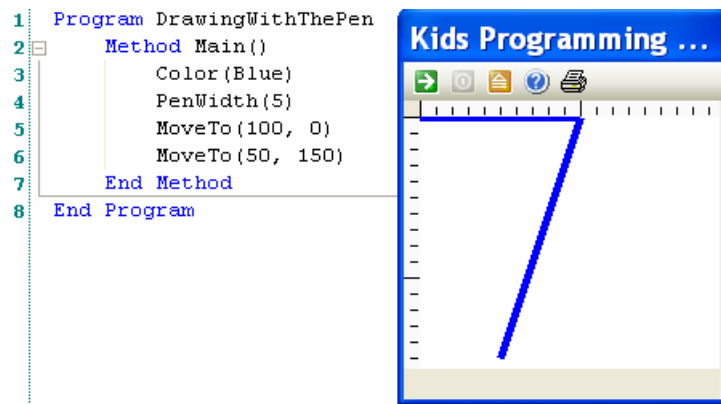
```
1 Program DrawingWithThePen
2 Method Main()
3
4 End Method
5 End Program
```

Nå kan vi begynne å legge instruksjoner inn i **Method Main()**, og se på hva hver av dem gjør:

```
1 Program DrawingWithThePen
2 Method Main()
3   Color(Blue)
4   PenWidth(5)
5 End Method
6 End Program
```



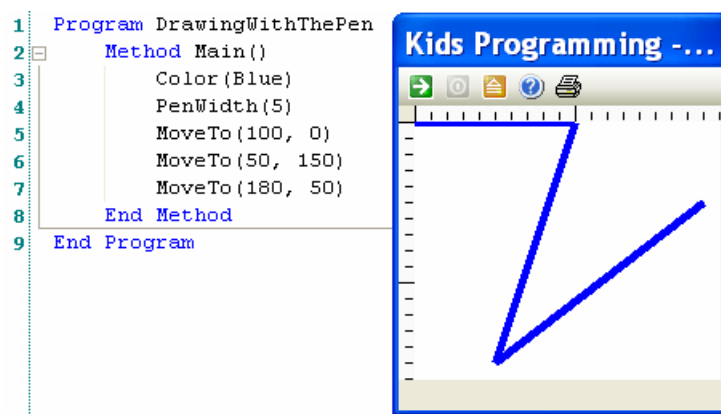
De to første instruksjonene er lagt inn i programmet, men når vi har kjørt det har ikke programmet tegnet opp noe grafikk ennå. **Color(Blue)** forteller KPL at vi vil tegne med blå farge. **PenWidth(5)** forteller KPL at når vi tegner med pennen, vil vi at KPL skal tegne en strek som er 5 bildepunkter bred. Du kunne selvfølgelig valgt en annen verdi for **PenWidth**. **PenWidth(2)** ville tegnet en tynnere strek, og **PenWidth(10)** ville tegnet en tykkere strek. Så nå har vi fortalt KPL hvordan det skal tegne, men vi har ikke sagt hva som skal tegnes ennå. Nå skal vi tegne den første streken i stjernen:



Den første instruksjonen vi legger til er **MoveTo (100, 0)** . Dette ber KPL om å flytte pennen til posisjonen (100, 0) på skjermen, som er toppen av stjernen. Siden KPL alltid starter med pennen i posisjonen (0, 0), det vil si øver venstre hjørne, vil du tegne den vannrette linjen du ser øverst i skjermbildet når du flytter pennen til posisjonen (100, 0).

Den andre instruksjonen vi legger til er **MoveTo (50, 150)** . Denne instruksjonen tegner den første streken i selve stjernen. Se et øyeblikk på (X, Y)-verdiene for disse to punktene. Siden denne instruksjonen ber KPL om å flytte pennen fra X = 100 til X = 50, beveger streken seg mot venstre. Og siden vi går fra Y = 0 til Y = 150, beveger streken seg nedover skjermen.

Nå skal vi legge inn enda en instruksjon, som legger til den neste streken i stjernen:

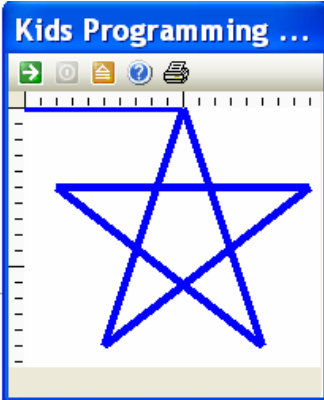


Instruksjonen vi legger til er **MoveTo (180, 50)** . KPL fortsetter å bevege pennen fra det forrige punktet, som var (50, 150). Har du sett leketøy som kalles Etch-A-Sketch, der du kan tegne på en slags skjerm ved å vri på to knotter? Pennen i KPL minner om en dataversjon av Etch-A-Sketch. Nå skal vi tegne resten av strekene i stjernen:


```

1 Program DrawingWithThePen
2   Method Main()
3     Color (Blue)
4     PenWidth(5)
5     MoveTo (100, 0)
6     MoveTo (50, 150)
7     MoveTo (180, 50)
8     MoveTo (20, 50)
9     MoveTo (150, 150)
10    MoveTo (100, 0)
11  End Method
12 End Program

```



Som du kan se, bruker vi ytterligere tre KPL-instruksjoner for å tegne tre streker til. Totalt har vi flyttet pennen seks ganger, og tegnet seks streker. Resultatet er en stjerne som er tegnet slik vi ba datamaskinen gjøre det i KPL-programmet.

KPL-programmet som skal til for å gjøre dette er lite, og består bare av 8 KPL-instruksjoner. Det vanskeligste med dette er å venne seg til hvordan (X, Y)-koordinatsystemet fungerer i KPL. Hvis du ikke forstår dette med (X, Y)-verdiene, anbefaler vi at du går tilbake til begynnelsen av dette kapitlet og leser det én gang til. Hvis du fremdeles ikke forstår hvordan (X, Y)-koordinatene fungerer, kan du forsøke å skrive din egen versjon av dette KPL-programmet i KPL på datamaskinen din, og eksperimentere med å tegne streker til og fra ulike posisjoner på skjermen. Forsøk å øve deg ved å tegne en firkant og en trekant. Hvis du gjør dette med KPL på datamaskinen, kan du hoppe til kapitlet **Bruke KPL på datamaskinen**, for å lære om praktisk bruk av KPL på datamaskinen. Når du føler deg komfortabel med (X, Y)-koordinatene, kan du fortsette herfra igjen.


Det er viktig at du bruker nok tid og innsats, slik at du føler at deg trygg på hvordan KPL bruker (X, Y)-koordinater, siden de er grunnlaget for all programmering av grafikk. Andre språk bruker samme koordinatsystem, så når du lærer dette i KPL, lærer du samtidig grunnlaget for programmering av grafikk i ethvert programmeringsspråk.

Nå skal vi legge til en avsluttende detalj i eksemplet vårt, som også viser en ekstra kontroll du har over pennen i KPL. Dette kan du ikke gjøre med en Etch-A-Sketch! Du kan heller ikke styre fargene på en Etch-A-Sketch, så la oss se hvor enkelt det er i KPL også:

```

1 Program DrawingWithThePen
2   Method Main()
3     Color (Green)
4     PenWidth(5)
5     Pen (False)
6     MoveTo (100, 0)
7     Pen (True)
8     MoveTo (50, 150)
9     MoveTo (180, 50)
10    MoveTo (20, 50)
11    MoveTo (150, 150)
12    MoveTo (100, 0)
13  End Method
14 End Program

```



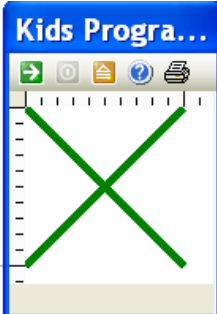
Som du ser endret vi nettopp fargen fra blå til grønn, ved å endre **Color (Blue)** til **Color (Green)**. Enkelt! Dessuten har vi lagt til instruksjonen **Pen (False)** like før **MoveTo (100, 0)**. **Pen (False)**

forteller KPL at pennen ikke skal tegne når den flyttes. Så nå når pennen beveger seg fra (0, 0) til (100, 0), tegner den ikke lenger den vannrette linjen frem til toppen av stjernen, og stjernen vår blir penere!

Husk at datamaskinen trenger nøyaktige og detaljerte instruksjoner, så da vi la inn instruksjonen **Pen (False)**, ba vi KPL om å ikke tegne når pennen beveger seg. Så hva skal vi gjøre når vi vil at KPL skal begynne å tegne strekene som stjernen består av? Da må vi gi KPL instruksjonen **Pen (True)**, slik at pennen begynner å tegne igjen når den beveger seg.

For å oppsummere: **Pen (False)** ber KPL om å slå "av" tegning når pennen flyttes, og **Pen (True)** ber KPL om å slå "på" tegning når pennen flyttes. Det kan ikke Etch-A-Sketch gjøre! Ved å slå pennen av og på mens du flytter den, kan du tegne alle slags gjenstander, og flere atskilte gjenstander. Her er et svært enkelt eksempel som du kan tegne ved å slå pennen av, flytte den, og deretter slå den på igjen:

```
1 Program DrawingWithThePen
2   Method Main()
3     Color (Green)
4     PenWidth (5)
5     MoveTo (100, 100)
6     Pen (False)
7     MoveTo (100, 0)
8     Pen (True)
9     MoveTo (0, 100)
10    End Method
11 End Program
```




Spillgrafikk med grafiske objekter

Poenget med tegneeksemplene over var hovedsaklig å forklare hvordan det grafiske (X, Y)-koordinatsystemet fungerer. Men grafikken i dataspill blir vanligvis ikke tegnet opp på denne måten. I stedet lastes den fra en bildefil, og kan for eksempel være et bilde av en UFO eller en asteroide eller en alv med en bue. KPL bruker grafiske objekter (kalt "sprite" på engelsk) for vise slik grafikk på en svært enkel måte.

Som i det forrige eksemplet, starter vi med å vise det ferdige programmet, og gir deretter en detaljert forklaring på hvordan programmet fungerer, slik at du kan skrive slike programmer selv. Her er hele KPL-programmet. Som du ser har det åtte instruksjoner inne **Method Main()**, akkurat som det forrige eksemplet. Men disse åtte instruksjonene ber KPL om å vise en UFO øverst i vinduet, og bevege den langsomt nedover skjermen til dit den er på figuren nedenfor:

```
1 Program UFO
2
3 Method Main()
4
5 LoadSprite( "UFO", "UFO.GIF" )
6 MoveSpriteToPoint( "UFO", 50, 0 )
7 ShowSprite( "UFO" )
8
9 Define ufoY As Int
10 For ufoY = 1 To 150
11 Delay(10)
12 MoveSpriteToPoint( "UFO", 50, ufoY )
13 Next
14
15 End Method
16
17 End Program
```



Legg merke til at dette programmet begynner og slutter på samme måte som våre tidligere eksempler, og som alle andre KPL-programmer, bortsett fra navnet, som i dette tilfellet er **UFO** :


```
1 Program UFO
2
3 Method Main()
4
5 End Method
6
7 End Program
```

Her er noen eksempler på bildefiler som følger med KPL. Du kan bruke en **hvilken som helst** bildefil du ønsker med KPL, inkludert filer du selv legger til eller lager (forutsatt at bildeformatet er riktig). Bildene under er bare et lite utvalg av de 65 bildene som følger med KPL:



Nå som du har sett noen eksempler på den grafikken du kan bruke med KPL, skal vi begynne med de KPL-instruksjonene som viser UFOen øverst på skjermen:

```
1 Program UFO
2
3 Method Main()
4
5     LoadSprite( "UFO", "UFO.gif" )
6     MoveSpriteToPoint( "UFO", 50, 0 )
7     ShowSprite( "UFO" )
8
9 End Method
10
11 End Program
```



Den første instruksjonen er `LoadSprite("UFO", "UFO.gif")`. Som nevnt må du følge visse regler når du gir KPL instruksjoner. Reglene for `LoadSprite` er ikke vanskelige, men du må følge dem nøye! `LoadSprite` må ha to "verdier" for å fungere. Den første verdien er navnet du velger på den grafiske figuren, som er "UFO" i vårt tilfelle. Den andre verdien er navnet på bildefilen som KPL henter figuren fra, som er "UFO.gif" her. Disse verdiene må stå i anførselstegn, som vist over. I tillegg må de skilles med et komma.

Her er noen eksempler på feil måte å laste denne grafiske figuren inn i KPL. For å gjenta: **Ingen av disse fire instruksjonene vil fungere**, siden de ikke følger reglene for `LoadSprite` i KPL. Klarer du å finne ut hva som må endres for å få dem til å fungere?

```
LoadSprite( 'UFO', 'UFO.gif' )
LoadSprite( UFO, UFO.gif )
LoadSprite( "UFO" . "UFO.gif" )
LoadSprite( "UFO" )
```

Oppsummering: `LoadSprite("UFO", "UFO.gif")` ber KPL om å lage en ny grafisk figur av bildefilen "UFO.gif", og kalle figuren "UFO".

Nå må KPL vite hvor på skjermen figuren skal plasseres, så vi legger til instruksjonen `MoveSpriteToPoint("UFO", 50, 0)`. `MoveSpriteToPoint` krever tre verdier. Det første er navnet på figuren. Som vi vet er det "UFO", siden det var navnet vi ga den med `LoadSprite`. Det andre er figurens **X-posisjon**, som vi vil skal være 50. Det tredje er figurens **Y-posisjon**, som vi vil skal være 0.


En viktig detalj du bør legge merke til er at **tall skal ikke stå i anførselstegn**. Generelt **krever** KPL anførselstegn rundt verdier som er **ord**, mens det **ikke** skal være anførselstegn rundt verdier som er **tall**. Se kapitlet om datatyper (Data Types) i brukerhåndboken for KPL (**KPL User Guide**) hvis du vil vite mer om dette.

Oppsummering: `MoveSpriteToPoint("UFO", 50, 0)` ber KPL om å flytte figuren med navnet "UFO" til skjermposisjonen (50, 0).

Det eneste som gjenstår nå er å be KPL om å vise den grafiske figuren. Det er den enkleste instruksjonen hittil: `ShowSprite("UFO")`. Der var det. Når du kjører programmet står UFOen på skjermen nøyaktig der vi ba KPL om å plassere den.

Det var en svært detaljert forklaring, så la oss gå tilbake til å vise hvor enkelt det faktisk er i KPL. Her er KPL-programmet, og hvordan det ser ut når vi kjører det.

```
1 Program UFO
2
3 Method Main()
4
5     LoadSprite( "UFO", "UFO.gif" )
6     MoveSpriteToPoint( "UFO", 50, 0 )
7     ShowSprite( "UFO" )
8
9 End Method
10
11 End Program
```



Bruke variabler og løkker i KPL

Nå vil vi at UFOen skal "lande" ved å bevege seg nedover skjermbildet til bunnen av vinduet. For å gjøre det skal vi definere og bruke vår første **"variabel"**, slik at du kan se hvordan variabler fungerer i et KPL-program. Du kan tenke på en variabel som en skuff i datamaskinen der vi lagrer en verdi.

Betegnelsen variabel er veldig beskrivende. Det betyr at verdien i skuffen kan endres. Nettopp dette er den store styrken til variabler, noe du vil se i dette eksemplet. Variabler bør defineres med et navn som beskriver hva de representerer og/eller hvordan de skal brukes. I dette tilfellet skal variabelen brukes til å endre UFOens verdi på Y-aksen, slik at den beveger seg nedover skjermen og "lander", så derfor har vi valgt å kalle variabelen **ufoY**.

Denne programlinjen oppretter variabelen:

```
Define ufoY As Int
```

Define er et KPL-nøkkelord som forteller KPL at du oppretter en ny variabel som KPL skal bruke. **ufoY** er navnet vi skal bruke på variabelen. **As Int** forteller KPL at **ufoY** defineres som en heltallsvariabel. **Int** er en forkortelse for "Integer", som betyr heltall. Heltallsvariabler kan inneholde hele tallverdier, for eksempel **-1** eller **0** eller **43**, **men ikke tall med komma**.

Vi vet at UFOen vår starter i posisjonen **(50, 0)** på skjermen, siden det var der vi ba KPL om å plassere den. Men hvordan får vi den til å bevege seg nedover skjermen? Det gjør vi ved å øke **Y-akseverdien**, til **(50, 1)** så **(50, 2)** så **(50, 3)** så **(50, 4)**, osv... Det er opp til oss hvor langt vi vil gå, så vi bestemmer oss for at UFOen stopper når den når **(50, 150)**.

Ser du mønsteret der? X-verdien for figurens posisjon er alltid **50**. Y-verdien for figurens posisjon øker med **ett** bildepunkt om ganger, fra **0** og opp til **150**. Det er viktig at du ser dette mønsteret før vi viser deg hvordan du bruker mønsteret et i KPL-program. Så hvis du ikke forstår mønsteret, bør du lese det siste avsnittet på nytt.

Her er programteksten som ber KPL om å øke verdien på variabelen **ufoY** fra 1 til 150, ett trinn om gangen:

```
For ufoY = 1 To 150
    MoveSpriteToPoint( "UFO", 50, ufoY )
Next
```

Dette er din første **løkke**, der en del av programmet kjøres om igjen flere ganger. Løkker er et annet av de nye konseptene du må lære hvis du ikke har programmert før. Men når du først har lært hvordan løkker fungerer, er det enkelt. Denne løkken starter med verdien **ufoY = 1**. Siden vi også har tatt med **To 150**, vet vi at løkken vil stoppe når den når verdien **ufoY = 150**. Legg merke til nøkkelordet **Next**, som definerer slutten på løkken. Når KPL kommer til nøkkelordet **Next**, **øker den verdien av ufoY** til neste verdi, fra 1 til 2, eller fra 2 til 3, eller fra 3 til 4, osv..., hele veien opp til 150.

I hovedsak ber vi KPL om å telle fra 1 til 150, og vi vil at KPL skal bruke vår variabel ufoY til å holde oversikt over verdien under tellingen.

Og hva vil vi at KPL-programmet skal gjøre mens det teller? Vi vil at det skal flytte UFOen nedover skjermen, ikke sant? Det er nettopp det **MoveSpriteToPoint ("UFO", 50, ufoY)** ber KPL om å gjøre.

Siden instruksjonen ligger inni **For**-løkken, vil KPL utføre denne instruksjonen **hver gang den teller** fra 1 til 150. Det er her mulighetene med løkker ligger. Bare å telle fra 1 til 150 er ikke så veldig interessant, men hvis du kan **gjøre noe nyttig hver gang du teller**, åpner det seg all slags morsomme og interessante muligheter, som for eksempel å flytte en UFO nedover skjermen. La oss se på hva som skjer:

```
For ufoY = 1 To 150
    MoveSpriteToPoint ( "UFO", 50, ufoY )
Next
```

Vi har allerede sett hvordan MoveSpriteToPoint fungerer. KPL flytter figuren **"UFO"** til den **(X, Y)**-posisjonen vi oppgir. Så hva er forskjellig denne gangen? Tidligere har vi flyttet UFOen til **(50, 0)**. Hva skjer når vi flytter den i denne løkken, og i stedet for et tall bruker variabelen **ufoY**? Husk at første gangen programmet går gjennom løkken, er verdien på **ufoY = 1**, og den andre gangen er **ufoY = 2**, så **ufoY = 3**, så **ufoY = 4**, og så videre... helt opp til **ufoY = 150**. Det betyr at den første gangen bruker programmet verdien **ufoY=1** til å flytte figuren, så instruksjonen som utføres er i praksis:

```
MoveSpriteToPoint ( "UFO", 50, 1 )
```

Og den neste gangen programmet går gjennom løkken er **ufoY = 2**, så da blir instruksjonen:

```
MoveSpriteToPoint ( "UFO", 50, 2 )
```

Slik fortsetter det, helt til programmet har talt seg gjennom alle tallene opp til 150:

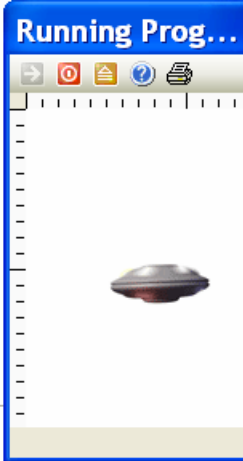
```
MoveSpriteToPoint ( "UFO", 50, 3 )
MoveSpriteToPoint ( "UFO", 50, 4 )
MoveSpriteToPoint ( "UFO", 50, 5 )
...
MoveSpriteToPoint ( "UFO", 50, 150 )
```

Hver gang figuren beveger seg, går den ett bildepunkt nedover skjermen, akkurat slik vi ville.

Det var altså din første variabel, din første løkke, og din første instruksjon inni en løkke. Alt dette er **svært viktige** konsepter innen programmering! Hvis du føler at du ikke forstår disse konseptene, anbefaler vi at du går tilbake til begynnelsen av dette kapitlet og leser det på nytt.

La oss legge til en detalj til for å fullføre programmet:

```
1 Program UFO
2
3 Method Main()
4
5     LoadSprite( "UFO", "UFO.GIF" )
6     MoveSpriteToPoint( "UFO", 50, 0 )
7     ShowSprite( "UFO" )
8
9
10    Define ufoY As Int
11    For ufoY = 1 To 150
12        Delay(10)
13        MoveSpriteToPoint( "UFO", 50, ufoY )
14    Next
15 End Method
16
17 End Program
```



Den eneste nye instruksjonen vi la til nå var **Delay (10)**, inni **For**-løkken. Hvorfor gjorde vi det? **Fordi datamaskiner teller veldig veldig raskt!** Tiden det tar for en datamaskin å telle fra 1 til 150 er kortere enn tiden det tar for meg å blinke med øynene. Helt sant! Så raske er datamaskinene! Hvis vi vil se UFOens bevegelse nedover skjermen, må vi bremse datamaskinens telling litt. **Delay (10)** ber ganske enkelt KPL-programmet om å ta en kort pause hver gang UFOen skal flyttes.

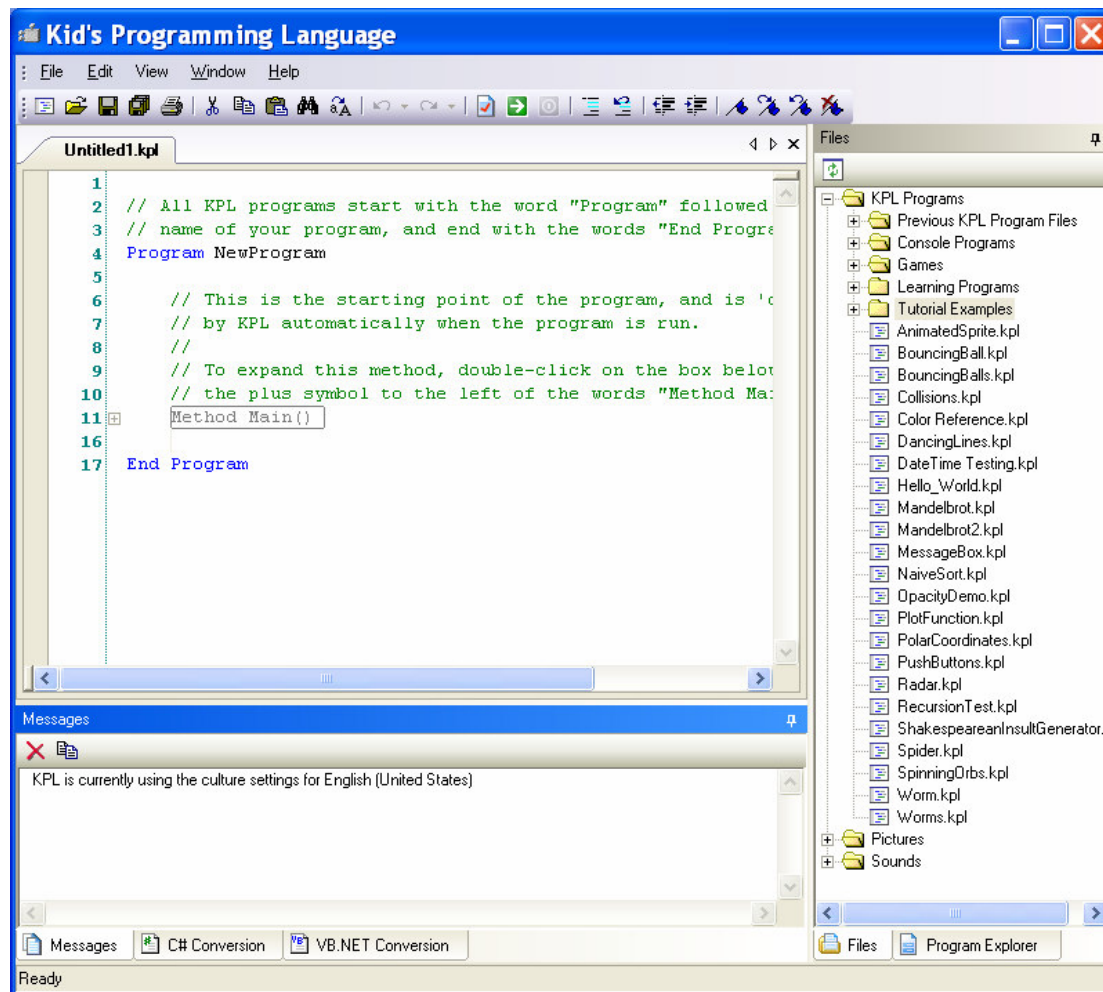
Du vet sikkert at når vi vil telle hvor mange sekunder som går, teller vi ved å si "tusen og én, tusen og to, tusen og tre", og så videre, for å telle passe langsomt. Dette er det samme prinsippet. Med **Delay (10)** gjør vi programmets telling langsommere. Når du arbeider med dette programmet selv i KPL, kan du forsøke å endre **10** til andre verdier, og så kjøre programmet. Du kan for eksempel prøve **Delay (2)**, og deretter **Delay (100)**. Det betyr ganske mye for hvordan UFOen beveger seg, ikke sant?

Vi har brukt flere sider på å forklare dette programmet i detalj, og derfor har du kanskje inntrykk av at det er mer komplisert enn det faktisk er. Men når du får litt taket på programmering i KPL, vil det ikke ta deg mange minuttene å skrive inn disse åtte instruksjonene, og forhåpentligvis vil du være enig i at bare noen få KPL-instruksjoner gjør det enkelt å lage kul grafikk med KPL!

Bruke KPL på datamaskinen

Viktig: Dette dokumentet forutsetter forbedringer i KPL som finnes i KPL-versjoner utgitt 10. oktober 2005 eller senere. Hvis du har lastet ned KPL før 10. oktober 2005, eller hvis eksemplene ikke fungerer som beskrevet når du skriver dem inn, bør du laste ned den nyeste versjonen av KPL fra <http://www.kidsprogramminglanguage.com/download.htm>.

Dette kapitlet forutsetter at KPL er installert på maskinen du bruker og at den engelske språkfilen er erstattet med den norske. Kapitlet legger vekt på å komme i gang med KPL. Finn frem til og start KPL fra oppføringen "Kids Programming Language" på startmenyen i Windows. Når du starter KPL, vil det se omtrent slik ut (men med norske menyer osv. hvis du har installert dette):

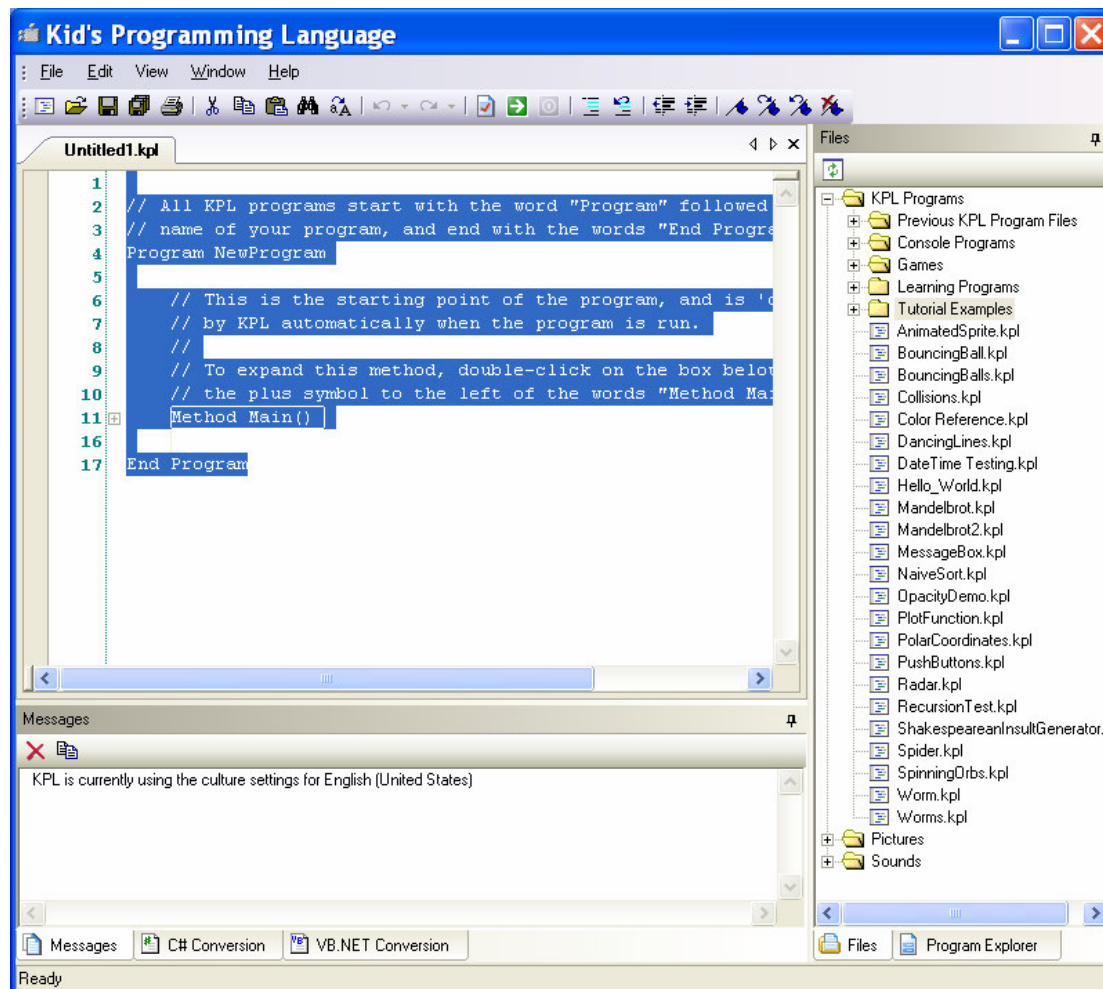


KPL viser en ny KPL-programfil med navnet Untitled1.kpl i et redigeringsvindu. De grønne linjene du ser i vinduet er "kommentarer", som gir informasjon til den som leser programmet, men som ikke inneholder noen aktive KPL-instruksjoner. Kommentarer er merket med to foroverlente skrånstreker: //

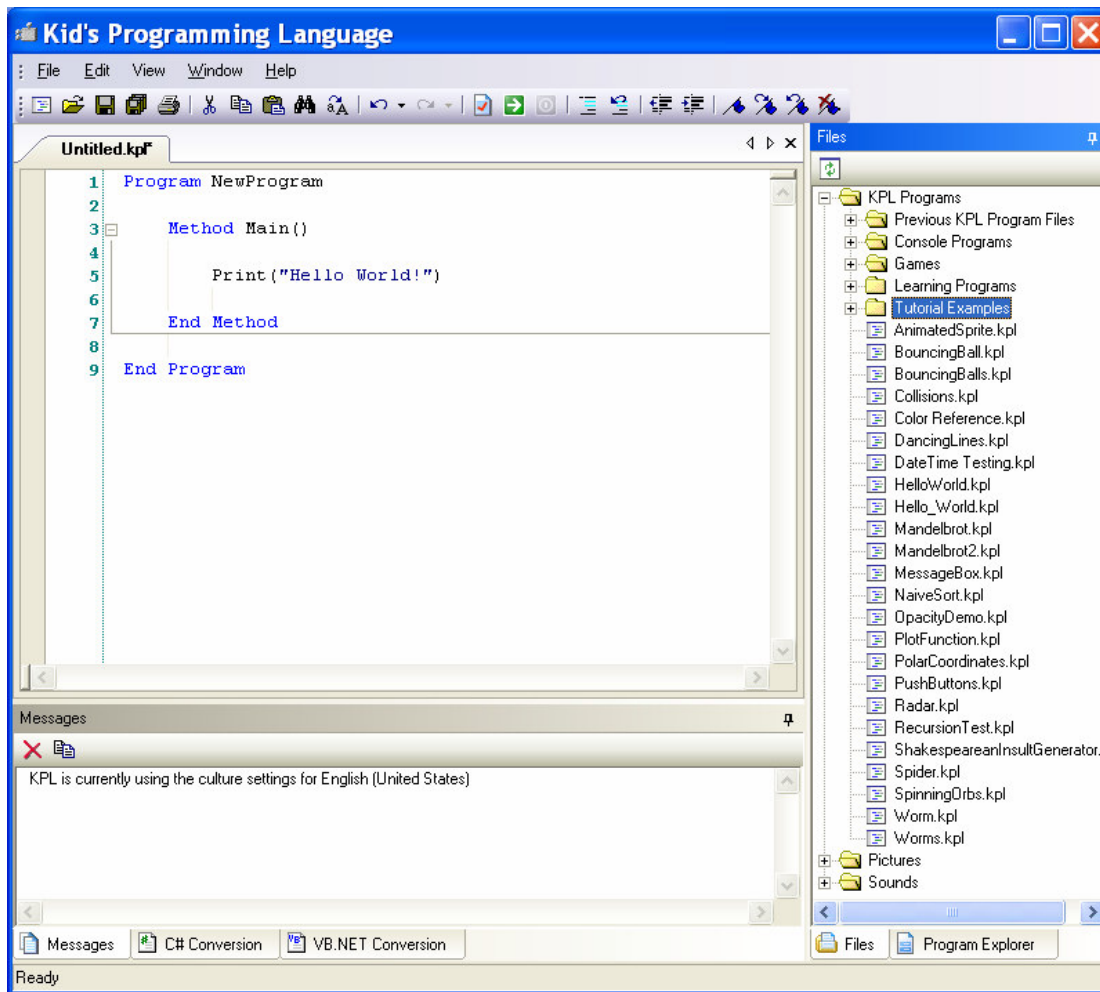
Du vil se mange kommentarer i eksempelprogrammene i KPL, og etter hvert vil du lære å bruke dem selv, når du skriver dine egne KPL-programmer. Kommentarer gjør det enklere for andre å forstå hva et KPL-program gjør. I tillegg kan de hjelpe deg med å huske detaljene i virkemåten til programmer du skrev for lenge siden.

Redigeringsvinduet i KPL har mange av de samme funksjonene som vanlige programmer for tekstbehandling og e-post. Se et øyeblikk på menyene og verktøylinjen. Når du holder musepekeren over verktøylinjeikonene, vil du se verktøytips som identifiserer dem og gir en kort forklaring.


Vi vil fokusere på å gjenskape de tre eksempelprogrammene vi har gått gjennom i denne innføringen, så la oss starte med å fjerne programteksten som vises i programfilen Untitled1.kpl. For å gjøre det, klikker du på **Rediger**-menyen og velger **Merk alt**. Du vil se at all programteksten i redigeringsvinduet nå er merket (uthevet), som vist nedenfor. Når all programteksten er merket på denne måten, kan du fjerne den enten ved å trykke **Delete**-tasten på tastaturet, eller ved å åpne **Rediger**-menyen og velge kommandoen **Klipp ut**. All merket tekst fjernes fra redigeringsvinduet i KPL.

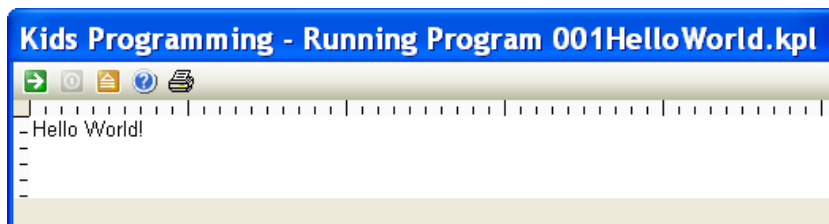


Når redigeringsvinduet er tomt begynner du å skrive inn ditt første KPL-program, slik det er vist her:



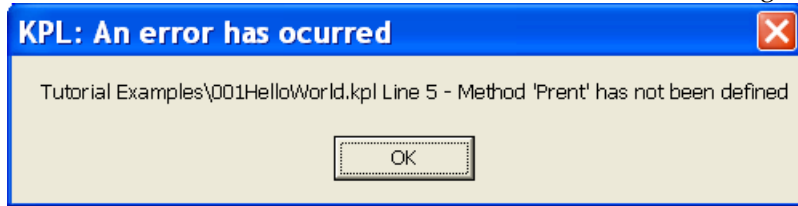
Husk at datamaskiner er svært nøyaktige og fantasiløse, så pass på å skrive KPL-programmet nøyaktig slik det står ovenfor. Hvis du forandrer på noe risikerer du at det ikke fungerer. Du kan bruke **TAB**-tasten til å rykke inn teksten som vist på figuren. Og du kan trykke **ENTER** for å lage blanke linjer. Det er ikke nødvendig å bruke innrykk og blanke linjer, men det vil gjøre det enklere for deg selv og andre å lese og forstå KPL-programmene dine.

Når du har skrevet inn programmet, kan du kjøre det ved å klikke på det grønne pilsymbolet  eller trykke **F5**-tasten. Hvis du har skrevet inn KPL-programmet uten feil, åpnes et vindu som ser ut som det nedenfor, bortsett fra at det vil være litt større:



Der var det. Nå er du en programmerer! Yesss!! Det er kanskje et svært lite og enkelt program, men du har uansett skrevet det inn og kjørt det selv.

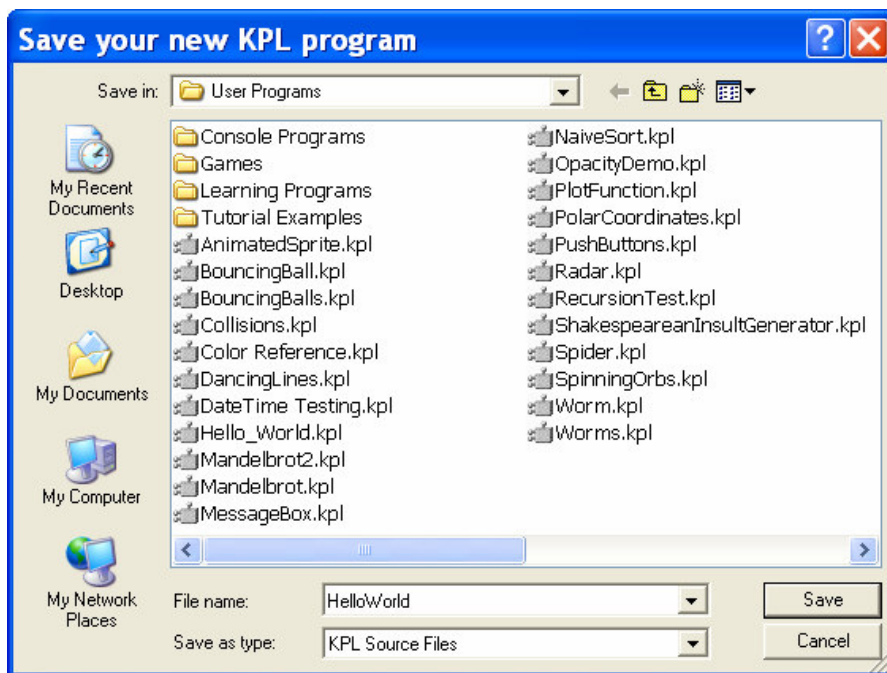
Hvis KPL ikke forstår det du har skrevet, vil du få en feilmelding tilsvarende den under.



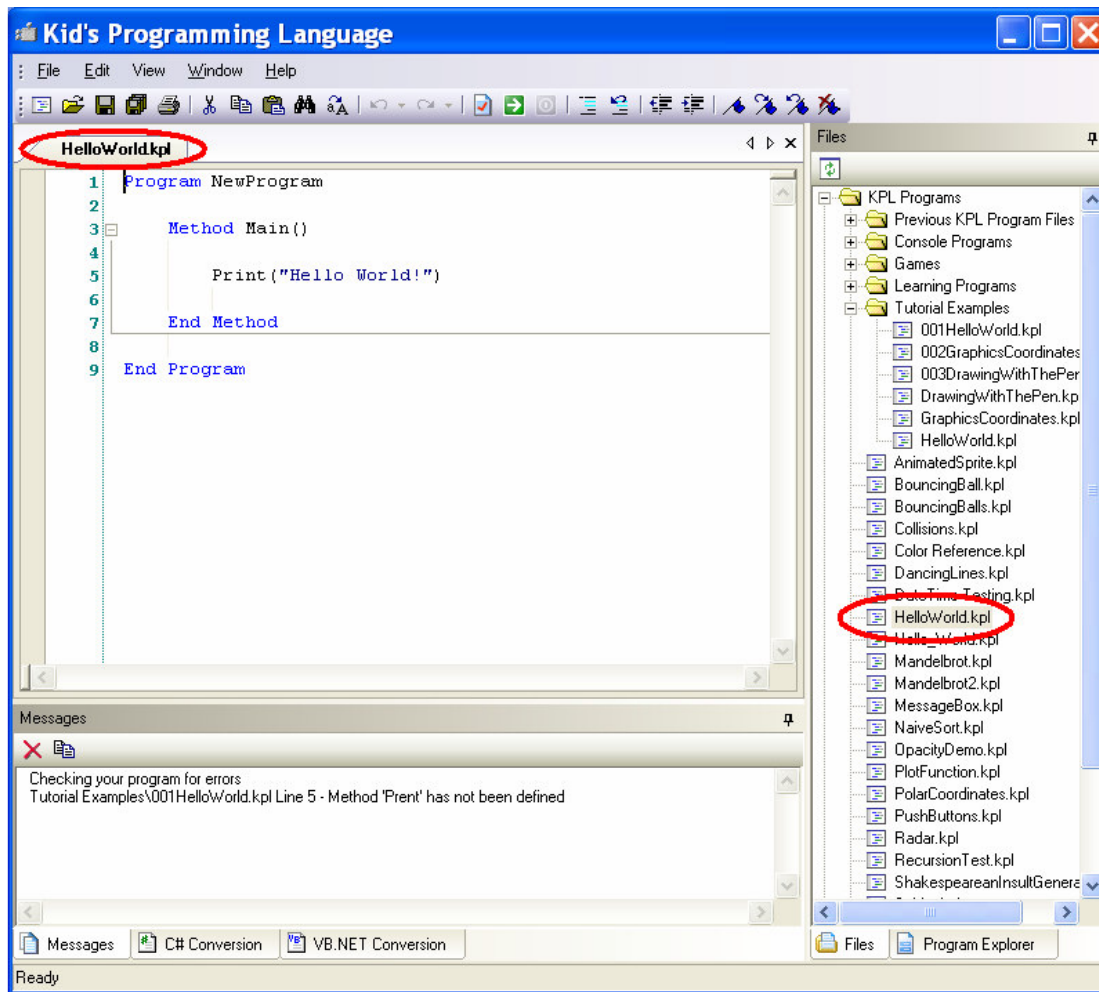
Hvis dette skjer, klikker du på OK. Deretter sammenligner du nøye programteksten du har skrevet inn mot skjermbildet ovenfor. Når du finner forskjellen og retter feilen slik at programteksten blir lik eksemplet her, vil du kunne kjøre programmet uten feil.

Selv de aller flinkeste programmererne har feil i programmene sine av og til, så du har ikke gjort noen dårlig jobb selv om du har noen feil. Ingen er perfekt! Hvis du har feil i programmet er det viktig at du forsøker å være rolig og tålmodig, og at du ser nøye gjennom det du har skrevet. Ro og tålmodighet er de beste våpnene mot feil, og gjør det enklere å finne ut hva som er feil og rette det.

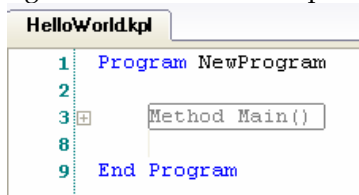
Når HelloWorld-programmet ditt fungerer, klikker du på **Fil**-menyen, velger **Lagre**, og gir programmet navnet HelloWorld som vist her:



Når du har lagret KPL-programmet, vil dette gjenspeile seg to steder, som vist her:

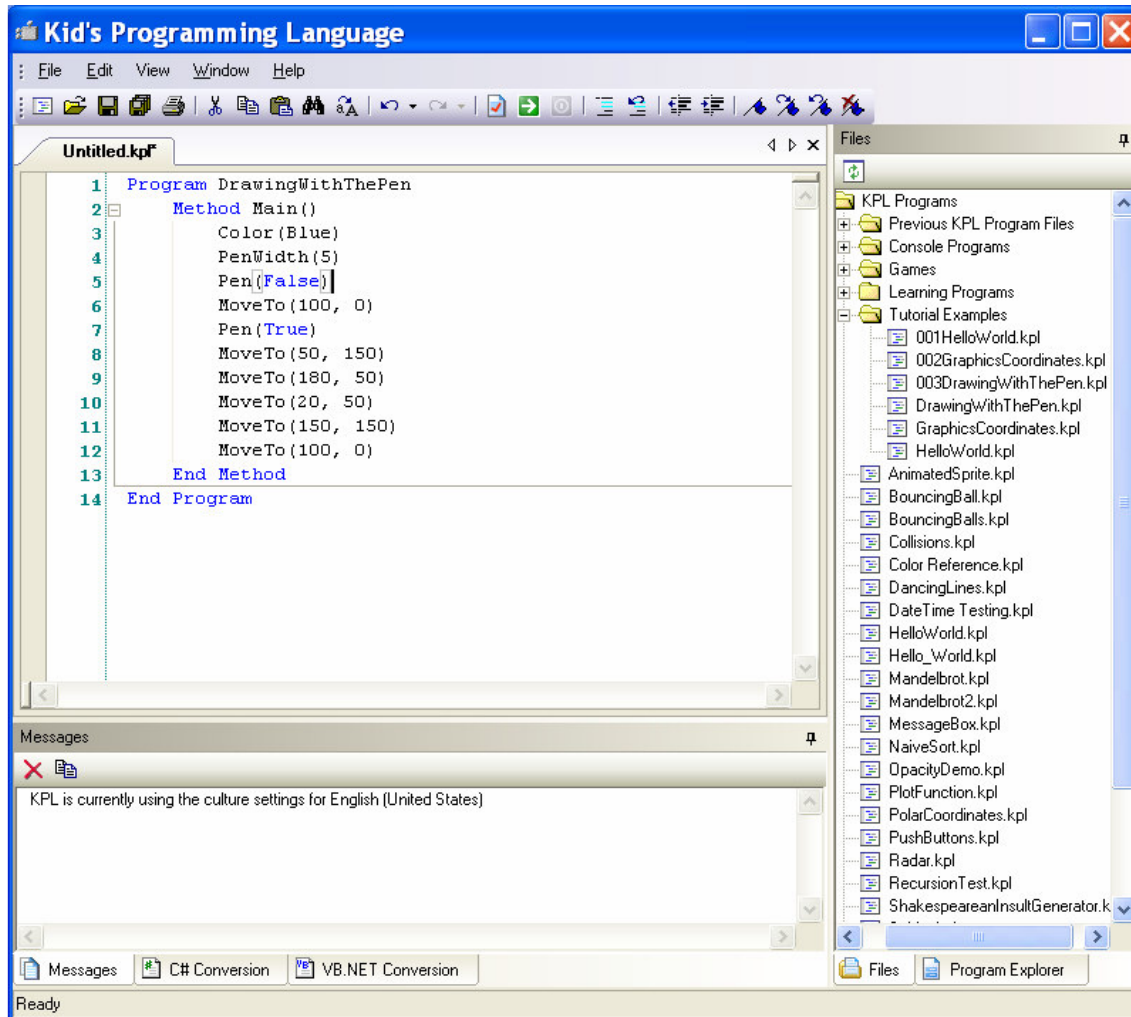


Nå som programmet er lagret på disken, kan du når som helst åpne det ved å dobbeltklikke på det i Filer-utforskeren til høyre i KPL. Når du åpner KPL-programmet ditt igjen senere, vil det trolig vises som på figuren nedenfor. Ikke få panikk, alt du skrev er der fremdeles!



Klikk på det lille plusstegnet (+) på linje 3, så vil all teksten i Method Main bli "åpnet" slik at du ser den igjen. Når du klikker på minustegnet (-), skjules teksten igjen. Dette er ikke særlig nyttig for et så enkelt og lite program som dette, men når du utvikler og skriver mye større programmer, vil du se at dette er en veldig nyttig og praktisk funksjon for deg som programmerer.

Når du har lagret **HelloWorld**, klikker du på **Fil**-menyen og velger **Nytt dokumentvindu**. Da kommer det opp et nytt **Untitled.kpl**-program, som det du startet med. Slett all tekst fra redigeringsvinduet på samme måte som forrige gang, og skriv inn programteksten for vårt andre eksempel:



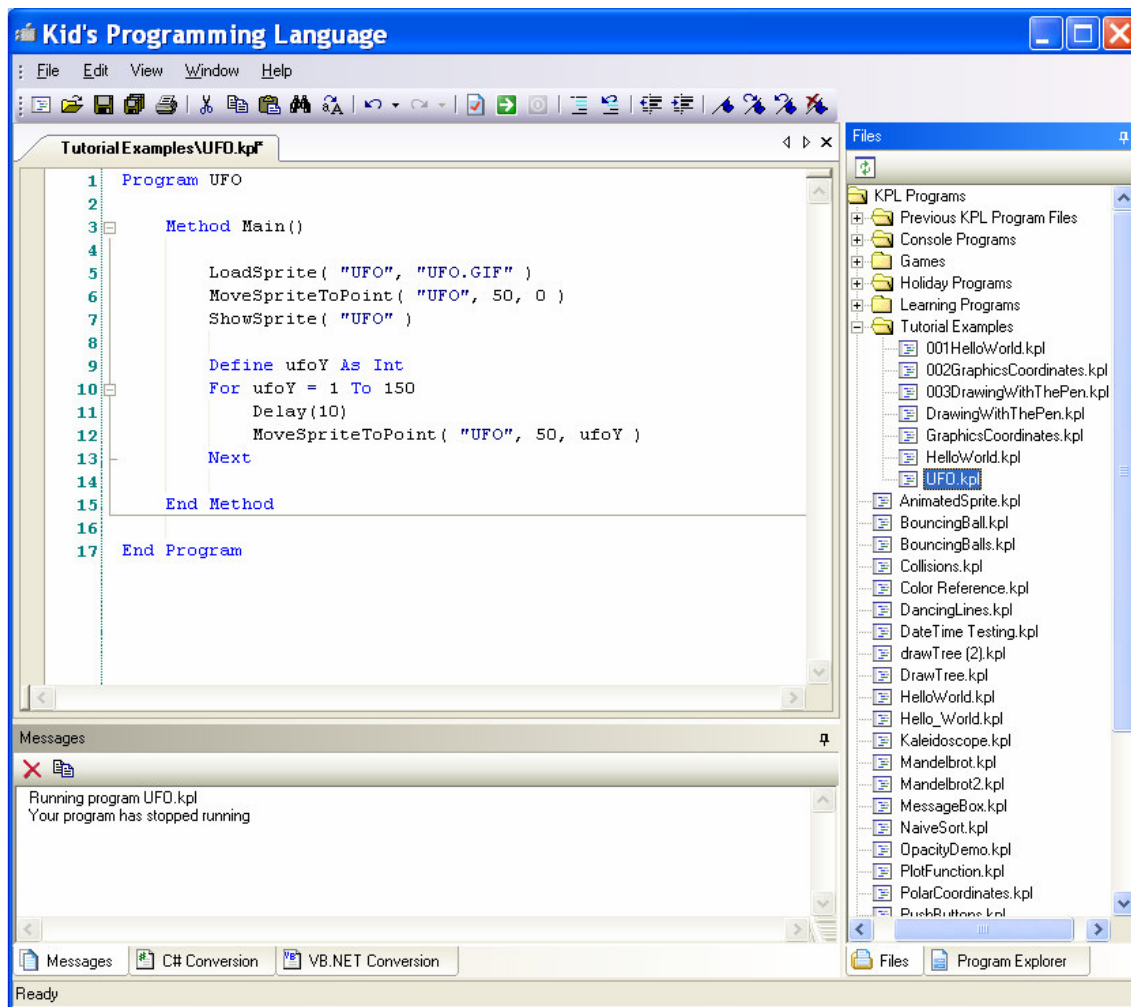
Skriver du inn programmet nøyaktig slik det står her og kjører det, vil du se den blå stjernen. Det betyr at du nå også er en grafisk programmerer. **Bra innsats!**

Husk at hvis det oppstår en feil, gjelder det å holde seg rolig, være tålmodig, og se nøye gjennom teksten for å finne forskjellen(e) mellom programmet ditt og dette eksemplet. Når du har endret teksten slik at den er helt lik figuren her, vil programmet fungere som det skal.

Husk å lagre programmet når du er ferdig med å skrive det!

Dette er også et morsomt program å eksperimentere med. Hvilke andre farger ser bra ut? Hvordan ser det ut hvis du ber pennen tegne tykkere eller tynnere streker? Hvordan ville du tegnet en firkant eller trekant i stedet for en stjerne? Og til slutt en nokså vanskelig oppgave: Hvordan ville du gjort stjernen større eller mindre?

Når du har lagret **DrawingWithThePen**, klikker du på **Fil**-menyen og velger **Nytt dokumentvindu**. Da kommer det opp et nytt **Untitled.kpl**-program. Slett all tekst fra redigeringsvinduet på samme måte som forrige gang, og skriv inn programteksten for UFO-eksemplet:



Hvis du skriver inn programmet nøyaktig slik det står her, og deretter kjører det, vil du se UFOen fly. **Bra!**

Husk at hvis det oppstår en feil, gjelder det å ta det rolig og se nøye gjennom teksten for å finne forskjellen(e) mellom programmet ditt og dette eksemplet. Når du har endret teksten din slik at den er helt lik figuren her, vil programmet fungere som det skal.

Husk å lagre programmet når du er ferdig med å skrive det!

Prøv disse morsomme øvelsene hvis du vil ha mer nyttig programmeringstrening:

- Kan du få UFOen til å bevege seg lengre?
- Hva med å få den til å fly fra venstre mot høyre, i stedet for ovenfra og ned?
- Klarer du å få UFOen til å fly nedover **og** mot høyre **samtidig**?

Neste trinn etter denne innføringen

Det første du bør gjøre når du er kommet så langt, er å finne mappen **Learning Programs** i KPL. Der finner du seks KPL-programmer som du kan fortsette å lære fra, og vi anbefaler at du går gjennom dem i rekkefølge (hvis du vil ha norske versjoner av eksempelprogrammene, finner du det på www.kat.no). De to første av dem er nokså lik programmer du allerede har gått gjennom i denne innføringen. Det er mulig at nye læreprogrammer vil være tilgjengelig når du leser dette. De vil i så fall enten være en del av KPL-installasjonen, eller så kan du finne dem på nedlastningssiden på www.kidsprogramminglanguage.com.

Det kan også være nyttig å laste ned vår brukerhåndbok for lærere (**User Guide for Teachers**), som du finner på den samme nedlastningssiden. Den er ikke ment som en innføring for begynnere, men kan være en nyttig referanse når du har lagt denne innføringen bak deg, og arbeider med dine egne KPL-programmer.

Når du har gjort deg kjent med alle læreprogrammene, kan du gå videre til noen av de mange andre eksempelprogrammene som følger med KPL, og åpne, utforske og lære av dem. **Kplong.kpl** og **NumberGuess.kpl** er spesielt nyttige i denne fasen. Du finner disse to programmene i **Games**-mappen. Dette er mye større programmer, og fullt funksjonelle spill. Men når du har forstått denne innføringen og alle læreprogrammene, vil du trolig ikke ha problemer med å forstå disse programmene heller.

Nedenfor finner du noen koblinger til diskusjons- og pratesider for KPL. Disse kan du også finne på siden med diskusjonsgrupper på www.kidsprogramminglanguage.com:

[Kule KPL-programmer](#)

[Barn diskuterer KPL](#)

[Foreldre diskuterer KPL](#)

[Lærere diskuterer KPL](#)

[Spørsmål og svar om KPL](#)

[Ulike språkversjoner av KPL](#)

Hvis du har spørsmål om KPL eller ønsker mer hjelp, er diskusjonsgruppen med spørsmål og svar om KPL ("KPL Questions and Answers") stedet å begynne. Disse diskusjonsgruppene er allerede nokså aktive, og aktiviteten øker stadig.