Microsoft Small Basic

Inngangur að forritun

Kafli 1

Kynning

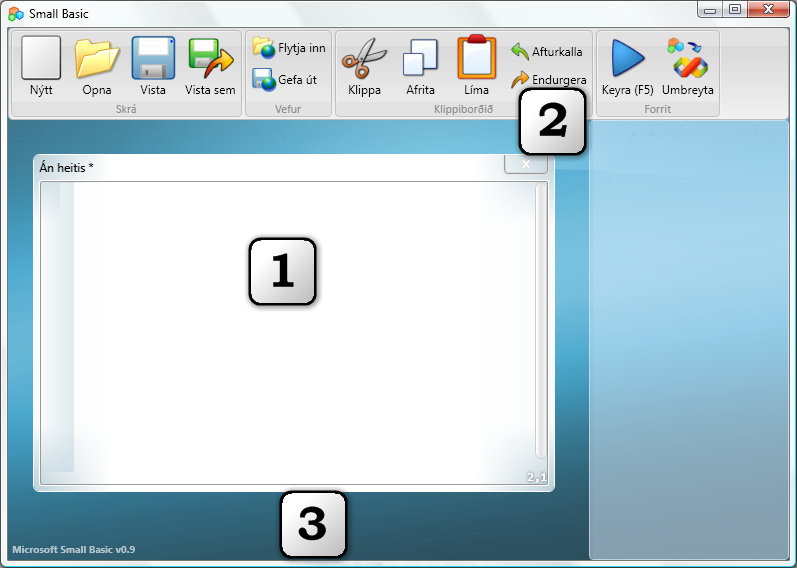
# Small Basic og forritun

Tölvuforritun er skilgreind sem gerð hugbúnaðar fyrir tölvur með því að nota forritunarmál. Alveg eins og við getum lesið og skilið tungumál svo sem íslensku eða ensku eða spænsku geta tölvur skilið forrit sem eru skrifuð á vissum tungumálum. Þau eru kölluð *forritunarmál*. Þegar tölvur komu fyrst fram voru aðeins til fá forritunarmál og það var mjög auðvelt að læra og skilja þau. En þegar tölvur og hugbúnaður varð fullkomnari, þróuðust forritunarmál hratt og bættu um leið við sig æ flóknari hugtökum. Afleiðingin er sú að í dag eru flest forritunarmál og hugtökin sem þau nota talsvert erfið fyrir byrjendur. Þessi staðreynd hefur orðið til að fæla fólk frá að læra eða prófa tölvuforritun.

Small Basic er forritunarmál sem er ætlað að gera forritun auðvelda, aðgengilega og skemmtilega fyrir byrjendur. Tilgangurinn með Small Basic er að lækka byrjunarþrepið og að þjóna sem leið inn í magnaðan heim tölvuforritunar.

# Small Basic umhverfið

Við skulum byrja á stuttri kynningu á Small Basic umhverfinu þar sem forrit eru búin til og prófuð. Þegar SmallBasic er keyrt í fyrsta sinn opnast gluggi sem líkist þeim sem er sést í myndinni hér á eftir.



Mynd 1 - Small Basic umhverfið

Þetta er Small Basic umhverfið þar sem við munum skrifa og keyra Small Basic forritin okkar. Í umhverfinu eru nokkur atriði sem hafa verið merkt með tölum í myndinni.

Merkt með [1] er **ritillinn** (enska: the Editor) þar sem Small Basic forritin eru skrifuð. Þegar þú opnar forrit, sem hefur verið vistað áður, birtist það í svona ritilglugga. Þá getur þú breytt því og vistað aftur að vild. Texti forrits er almennt kallaður *kóði* eða *forritskóði* (enska: code).

Það er hægt að opna og vinna með fleiri en eitt forrit í einu. Hvert forrit er þá í sjálfstæðum ritilglugga. Ritillinn sem inniheldur forritið sem þú ert að vinna í þá stundina nefnist *virki ritillinn*.

**Tækjastikan**, sem er merkt með [2], er notuð til að gefa skipanir - annaðhvort fyrir virka ritilinn eða fyrir umhverfið almennt. Við munum læra um hinar ýmsu skipanir eftir þörfum jafnóðum.

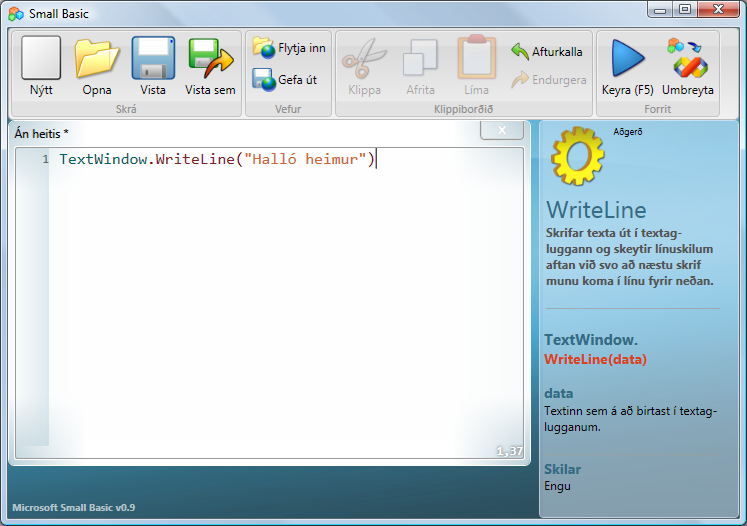
**Bakgrunnurinn**, sem er merktur með [3], er staðurinn þar sem allir opnir ritilgluggar eru hafðir.

# Fyrsta forritið okkar

Nú hefur þú kynnst Small Basic umhverfinu og kominn tími til að byrja að forrita í því. Eins og áður var sagt skrifum við forritin í ritlinum. Svo við skulum drífa í að skrifa eftirfarandi línu í ritilinn.

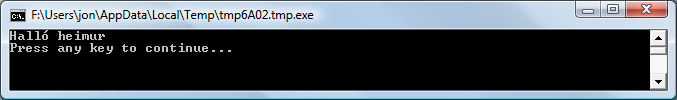
TextWindow.WriteLine("Halló heimur")

Þetta er fyrsta Small Basic forritið okkar. Og ef þú hefur skrifað það rétt, ættir þú að sjá eitthvað svipað því sem myndin hér fyrir neðan sýnir.



Mynd 2 – Fyrsta forritið

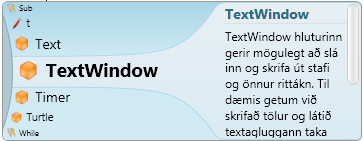
Þetta var allt og sumt! Og nú skulum við keyra það til að sjá hvað það gerir. Við getum keyrt forrit annaðhvort með því að smella á hnappinn *Keyra* á tækjastikunni eða með því að nota flýtilykil, *F5*, á lyklaborðinu. Ef allt gengur vel, keyrir forritið og gefur útkomuna sem sést hér fyrir neðan. Ef ekki þarftu að athuga hvort það er rétt skrifað og prófa aftur.



Mynd 3 – Fyrsta útkoman úr forriti

Til hamingju! Þú hefur nú skrifað og keyrt fyrsta Small Basic forritið. Reyndar ósköp lítið og einfalt forrit, en samt stórt skref í þá átt að verða alvöru tölvuforritari! Nú er bara eitt smáatriði sem við þurfum að fara í áður en við getum byrjað að skrifa stærri forrit. Við þurfum að skilja hvað það var sem gerðist – nákvæmlega hvað það var sem við sögðum tölvunni og hvernig tölvan vissi hvað hún átti að gera. Í næsta kafla munum við skoða forritið sem við vorum að skrifa til að fá þennan skilning.

*Þegar þú varst að skrifa fyrsta forritið, hefurðu áreiðanlega tekið eftir lista sem birtist (mynd 4). Þessi tækni er kölluð "intellisense" (eiginlega gáfuleg skynjun) og hún er til að hjálpa þér að skrifa forritið hraðar og réttar. Þú getur flett í listanum með því að nota lyklana NIÐURÖR og UPPÖR. Þegar þú finnur það sem þú vilt nota, getur þú slegið á FÆRSLULYKILINN til að setja valda atriðið inn í forritið.*



Mynd 4 – Intellisense hjálpin

# Forritið vistað

Ef þú vilt loka Small Basic og halda áfram seinna með forrit sem þú ert að skrifa, getur þú vistað forritið. Það er reyndar góð vinnuregla að vista forrit sem þú ert að vinna í af og til, svo að þú glatir ekki upplýsingum ef forritið skyldi lokast óvænt eða rafmagnið fara af. Þú getur vistað forritið í virka ritlinum með því smella á hnappana *Vista* eða *Vista sem* á tækjastikunni eða með því að nota flýtilyklana *CTRL + S* (sem táknar að það eigi að halda niðri CTRL lykli og slá á S).

Kafli 2

Að skilja fyrsta forritið

# Hvað í raun og veru er tölvuforrit?

Forrit er safn fyrirmæla sem tölvan á að fara eftir. Þessi fyrirmæli segja tölvunni nákvæmlega hvað hún á að gera og tölvan fylgir alltaf þessum fyrirmælum í blindni. En eins og fólk geta tölvur aðeins fylgt leiðbeiningum ef þær eru á tungumáli sem þær skilja. Tungumál fyrir tölvur eru kölluð forritunarmál. Tölvan skilur mjög mörg forritunarmál og Small Basic er eitt þeirra.

Ímyndaðu þér samtal sem á sér stað milli þín og vinar þíns. Þið notið orð sem er raðað saman í setningar til að bera upplýsingar fram og til baka á milli ykkar. Á sama hátt eru forritunarmál söfn af orðum sem er hægt að raða í setningar sem flytja tölvunni upplýsingar. Og forrit eru í grunninn safn setninga (stundum bara örfárra og stundum fleiri þúsunda) sem saman hafa einhverja merkingu bæði fyrir forritarann og tölvuna.

*Það eru mörg tungumál sem tölvan skilur. Java, C++, Python, VB, o.fl. eru öll öflug nútíma forritunarmál sem eru notuð til að þróa bæði einfaldan og flókinn hugbúnað.*

# Small Basic forrit

Dæmigert Small Basic forrit samanstendur af safni *setninga*. Hver lína í forritinu er ein setning og hver setning er fyrirmæli til tölvunnar. Þegar við biðjum tölvuna að keyra Small Basic forrit tekur hún forritið og les fyrstu setninguna. Ef setningin er rétt skrifuð, skilur tölvan hvað við erum að biðja hana að gera og framkvæmir það. Eftir að hún er búin að framkvæma fyrirmælin í fyrstu setningunni færir hún sig niður í næstu línu og les fyrirmælin þar og framkvæmir þau. Svona heldur hún áfram þar til hún finnur ekki fleiri línur til að lesa. Þar með endar forritið.

# Aftur að fyrsta forritinu

Hér er fyrsta forritið sem við skrifuðum:

TextWindow.WriteLine("Halló heimur")

Þetta er mjög einfalt forrit sem inniheldur eina *setningu*. Þessi setning segir tölvunni að skrifa línu af texta sem er **Halló heimur** í textagluggann.

Tölvan túlkar þetta bókstaflega sem:

Skrifaðu **Halló heimur**

Þú hefur kannske þegar tekið eftir að það er hægt að skipta setningunni upp í smærri hluta alveg eins og við getum skipt okkar setningum í orð. Fyrsta setningin er greinilega sett saman úr 3 hlutum:

1. TextWindow
2. WriteLine
3. “Halló heimur”

Punkturinn, sviginn og gæsalappirnar ("") eru allt greinarmerki sem verða að vera á réttum stað í setningunni ef tölvan á að geta skilið hvað við ætlumst til af henni.

Þú manst ef til vill eftir svarta glugganum sem birtist þegar við keyrðum fyrsta forritið. Þessi svarti gluggi er kallaður textaglugginn en er stundum nefndur skipanalínan. Útkoman úr forritinu birtist þar. **TextWindow** sem við notuðum í forritinu okkar er dæmi um *hlut* (object). Þegar við skrifum forrit getum við notað þó nokkuð marga svona hluti. Hver hlutur hefur mismunandi *aðgerðir* sem við notum til að vinna með hlutinn. TextWindow hluturinn hefur margar aðgerðir. Við höfum þegar notað *WriteLine* aðgerðina í forritinu okkar.

*Greinarmerki svo sem gæsalappir, bil og svigar eru mjög mikilvæg í tölvuforriti. Staðsetning þeirra og fjöldi getur breytt merkingu þess sem er skrifað.*

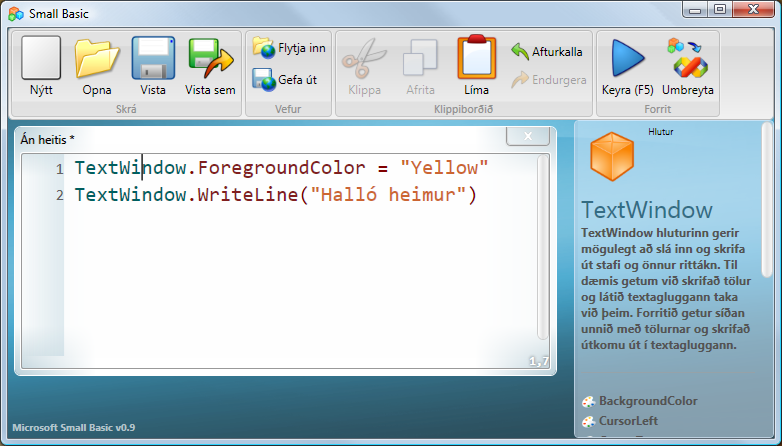
Þú hefur ef til vill líka tekið eftir því að á eftir WriteLine aðgerðinni fylgir **Halló heimur** innan gæsalappa. Þessi texti er tekin inn í WriteLine aðgerðina, sem síðan skrifar hana út svo notandinn sjái hann. Þetta er kallað *færibreyta* aðgerðarinnar (enska: parameter). Sumar aðgerðir hafa eina eða fleiri færibreytur en aðrar hafa enga.

# Annað forrit

Nú hefurðu fengið skilning á fyrsta forritinu svo það er kominn tími til að endurbæta það með litum.

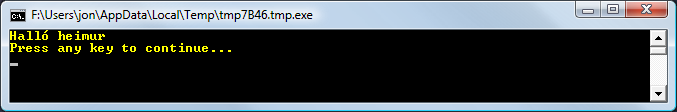
TextWindow.ForegroundColor = "Yellow"

TextWindow.WriteLine("Halló heimur")



Mynd 5 – Lit bætt við

Þegar þú keyrir forritið sést að það skrifar út eins og áður "Hello heimur" í textaglugganum, en í þetta sinn er textinn skrifaður út í gulum lit en ekki gráum eins og áður.



Mynd 6 – Halló heimur í gulu

Taktu eftir nýju setningunni sem við bættum við upphaflega forritið. Hún notar nýtt orð, *ForeGroundColor* sem við gerðum jafnt og *"Yellow"* eða gult. Við höfum sett gildið "Yellow" sem innihald í ForeGroundColor. Munurinn á ForeGroundColor og aðgerðinni WriteLine er að ForeGroundColor hefur enga færibreytu og þarf ekki sviga. Í staðinn var notað jafnaðarmerki (=) og heiti á lit. ForeGroundColor er dæmi um *eiginleika* (enska: property) sem TextWindow hluturinn hefur. Hér fyrir neðan er listi yfir gildi sem má nota fyrir ForeGroundColor eiginleikann. Prófaðu að nota eitt þessara orða í stað "Yellow" en gleymdu ekki gæsalöppunum – þær eru nauðsynlegar.

|  |  |
| --- | --- |
| Black  Blue  Cyan  Gray  Green  Magenta  Red  White | *Yellow*  *DarkBlue*  *DarkCyan*  *DarkGray*  *DarkGreen*  *DarkMagenta*  *DarkRed*  *DarkYellow* |

Kafli 3

Kynning á breytum

# Að nota breytur í forriti

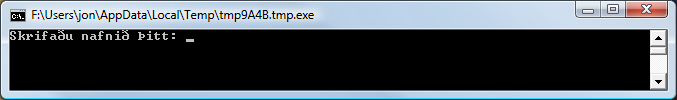
Hvernig væri ef forritið okkar gæti sagt "Halló" og látið nafn notandans fylgja í stað þess að nota bara almennt "Halló heimur"? Til að það sé hægt verðum við fyrst að spyrja notandann hvað hún/hann heitir og geyma þær upplýsingar einhverstaðar þangað til forritið getur skrifað út "Halló" með nafni notandans. Við getum gert það svona:

TextWindow.Write("Skrifaðu nafnið þitt: ")

nafn = TextWindow.Read()

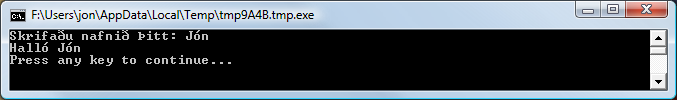
TextWindow.WriteLine("Halló " + nafn)

Þegar þú skrifar og keyrir þetta forrit sérðu útkomu sem líkist eftirfarandi:



Mynd 7 – Spurt um nafn notandans

Og þegar þú skrifar nafn þitt og slærð á FÆRSLULYKILINN, sérðu útkomu sem líkist þessu:



Mynd 8 – Hlýleg kveðja

Þegar forritið er keyrt aftur, spyr forritið sömu spurningar aftur. Nú getur þú skrifað annað nafn og tölvan notar þá það nafn í kveðjunni.

# Greining á forritinu

Í forritinu sem þú varst að keyra hefur þú ef til vill tekið eftir þessari línu:

nafn = TextWindow.Read()

*Read()* er svipað og *WriteLine()*, en án færibreytu. Þetta er aðgerð og hún segir tölvunni að bíða á meðan notandinn skrifar eitthvað og þangað til hann slær á FÆRSLULYKILINN. Eftir að notandinn slær á FÆRSLULYKILINN tekur aðgerðin það sem notandinn hefur skrifað og skilar því inn í forritið. Það áhugaverðasta við þetta er að það sem notandinn skrifaði er nú sett til geymslu í *breytu* (enska: variable) sem fær heitið **nafn**. *Breyta* er skilgreind sem staður eða hólf þar sem við getum geymt gildi (t.d. tölur eða orð) tímabundið til notkunar seinna. Í línunni fyrir ofan var breytan **nafn** notuð til að geyma nafn notandans.

Næsta lína er einnig áhugaverð:

TextWindow.WriteLine("Halló " + nafn)

Það er hér sem við notum gildið sem var geymt í breytunni **nafn**. Við tökum gildið í **nafn** og skeytum því aftan við "Halló" (strangt tekið   
"Halló ") og skrifum það út í textagluggann.

Write er önnur aðgerð svipuð WriteLine í TextWindow. Með Write er hægt að skrifa út í textagluggann þannig að næsta útskrift komi á sömu línu og það sem er skrifað með Write.

Þegar búið er að skilgreina breytu er hægt að nota hana aftur og aftur. Til dæmis gætum við gert eftirfarandi:

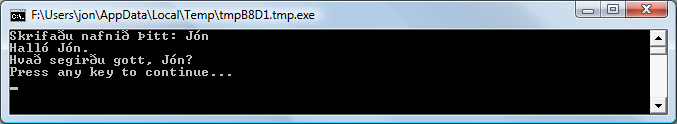
TextWindow.Write("Skrifaðu nafnið þitt: ")

nafn = TextWindow.Read()

TextWindow.Write("Hello " + **nafn** + ". ")

TextWindow.WriteLine("Hvað segirðu gott, " + **nafn** + "?")

Þetta gefur útkomu sem líkist þessu:



Mynd 9 – Breyta endurnýtt

# Reglur um heiti breytna

Hver breyta hefur heiti og það er þannig sem við þekkjum þær sundur og vitum hvað er geymt í þeim. Um það hvað breytur mega heita og hvað þær ættu að heita eru hér nokkrar einfaldar reglur og góðar leiðbeiningar:

1. Heitið verður að byrja á bókstaf og má ekki vera eitt af lykilorðum Small Basic svo sem **if**, **for**, **then** o.s.frv. (meira um þau seinna).
2. Heitið má aðeins innihalda bókstafi, tölustafi og lágstrikið ( \_ ) en ekki önnur tákn.
3. Það er ekki nauðsynlegt en mjög gagnlegt að gefa breytum lýsandi heiti – lengd heitanna er ekki takmörkuð svo að það er auðvelt að láta sjást á heitinu til hvers breyta er notuð.

# Leikið að tölum

Við höfum séð hvernig við gátum notað breytu til að geyma nafn notandans. Í næstu forritum munum við skoða hvernig við getum geymt og unnið með tölur sem geymdar eru í breytum. Við skulum byrja á mjög einföldu forriti:

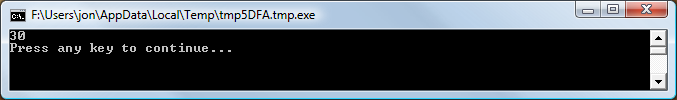
tala1 = 10

tala2 = 20

tala3 = tala1 + tala2

TextWindow.WriteLine(tala3)

Þegar þetta forrit er keyrt fáum við eftirfarandi útkomu:



Mynd 10 – Tvær tölur lagðar saman

Í fyrstu línu forritsins gefum við breytunni **tala1** gildið 10. Í annarri línu gefum við breytunni **tala2** gildið 20. Í þriðju línunni leggjum við saman innihaldið í **tala1** og **tala2** og setjum útkomuna í **tala3**. Þannig að nú hefur breytan **tala3** innihaldið 30. Og það er það sem er skrifað út í textagluggann.

*Taktu eftir að tölurnar hafa ekki gæsalappir utan um sig. Þær eru ekki nauðsynlegar fyrir tölur. Það er aðeins texti sem þarf að vera afmarkaður með gæsalöppum.*

Nú skulum við breyta forritinu aðeins og skoða útkomuna:

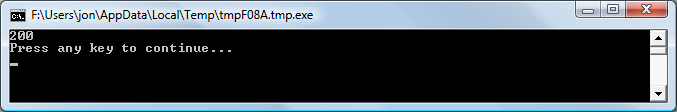
tala1 = 10

tala2 = 20

tala3 = tala1 \* tala2

TextWindow.WriteLine(tala3)

Forritið hér fyrir ofan margfaldar **tala1** með **tala2** og geymir útkomuna í **tala3**. Og útkoman úr forritinu sést hér fyrir neðan:



Mynd 11 – Tvær tölur margfaldaðar

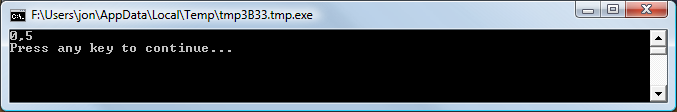
Á sama hátt má draga frá eða deila tölum. Hér er frádrátturinn:

tala3 = tala1 - tala2

Og táknið fyrir deilingu er / og forritið lítur þá svona út:

tala3 = tala1 / tala2

Og útkoman úr þessari deilingu yrði:



Mynd 12 – Deiling talna

# Einfaldur hitastigsreiknir

Í næsta forriti munum við nota formúluna til að umreikna hitastig í gráðum Fahrenheit (sem eru notaðar t.d. í Bandaríkjunum ) yfir í gráður Celsius (sem eru notaðar t.d. á Íslandi).

Fyrst biðjum við notandann um hitastigið í Fahrenheit og geymum það í breytu. Það er til sérstök aðgerð, **TextWindow.ReadNumber**, sem les eingöngu tölur sem notandinn skrifar.

TextWindow.Write("Sláðu inn hitastig í Fahrenheit gráðum: ")

fahr = TextWindow.ReadNumber()

Eftir að við höfum sett Fahrenheit hitastigið í breytu, getum við umreiknað það yfir í Celsius þannig:

celsius = 5 \* (fahr - 32) / 9

Sviginn segir tölvunni að reikna **fahr-32** hlutann fyrst og nota þá útkomu í framhaldinu. Nú þurfum við aðeins að skrifa út svarið fyrir notandann. Þegar þetta er allt sett saman fáum við þetta forrit:

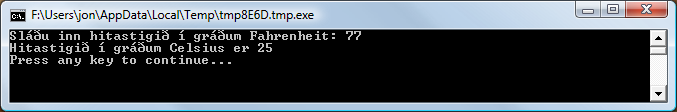
TextWindow.Write("Sláðu inn hitastigið í gráðum Fahrenheit: ")

fahr = TextWindow.ReadNumber()

celsius = 5 \* (fahr - 32) / 9

TextWindow.WriteLine("Hitastigið í gráðum Celsius er " + celsius)

Og útkoman úr forritinu gæti verið svona:



Mynd 13 – Umreiknun hitastigs

Kafli 4

Skilyrði og greinar

Ef við snúum okkur aftur að fyrsta forritinu, væri ekki sniðugt ef við gætum látið það segja *Góðan daginn heimur* eða *Gott kvöld heimur* á mismunandi tímum dags í stað þess að segja bara alltaf *Halló heimur*? Í næsta forriti ætlum við að láta tölvuna segja *Góðan daginn heimur* ef klukkan er ekki orðin 18 en *Gott kvöld heimur* ef klukkan er 18 eða meira.

If (Clock.Hour < 18) Then

TextWindow.WriteLine("Góðan daginn heimur")

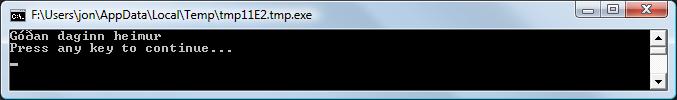
EndIf

If (Clock.Hour >= 18) Then

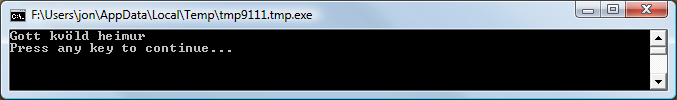
TextWindow.WriteLine("Gott kvöld heimur")

EndIf

Eftir því hvenær forritið er keyrt gæti útkoman verið önnur hvor af þessum:



Mynd 14 – Góðan daginn heimur



Mynd 15 – Gott kvöld heimur

Við skulum skoða fyrstu þrjár línur forritsins. Þú hefur líklega þegar áttað þig á því að þessar línur segja tölvunni að ef klukkustundin fengin með Clock.Hour er minni en 18 þá verði skrifað út "Góðan daginn heimur". Orðin **If**, **Then** og **EndIf** eru sérstök lykilorð (enska: keywords) sem tölvan skilur. Á eftir orðinu **If** (íslenska: ef) fylgir alltaf skilyrði, sem í þessu tilfelli er **(Clock.Hour < 18)**. Mundu að svigarnir eru nauðsynlegir ef tölvan á að geta skilið til hvers þú ætlast af henni. Fyrir aftan skilyrðið kemur orðið **then** (íslenska: þá) og síðan í næstu línu fyrsta aðgerðin sem á að framkvæma ef skilyrðið er uppfyllt. Síðast kemur **EndIf** sem segir tölvunni að skilyrðin gildi ekki lengur fyrir skipanir sem á eftir koma.

*Í Small Basic, getur þú notað Clock hlutinn til að finna dagsetninguna og tímann. Hann hefur einnig fjölda eiginleika sem gerir mögulegt að finna út mánaðardaginn, mánuðinn, árið, klukkustundina, mínútuna og jafnvel sekúnduna hvert fyrir sig.*

Á milli **then** og **EndIf** geta verið margar skipanir og tölvan mun framkvæma þær allar ef skilyrðið er uppfyllt (en sleppa þeim annars). Þú gætir til dæmis skrifað eitthvað þessu líkt:

If (Clock.Hour < 18) Then

TextWindow.Write("Góðan daginn. ")

TextWindow.WriteLine("Hvernig er veðrið?")

EndIf

# Else

Í forritinu í upphafi kaflans hefur þú ef til vill tekið eftir því að seinna skilyrðið er eiginlega óþarft. Gildið sem Clock.Hour gefur (þ.e. klukkustund sólarhringsins) er annaðhvort minna en 18 eða ekki. Við þurftum í raun ekki að athuga tímann aftur. Þegar við höfum svona tvö skilyrði þar sem annað útilokar hitt, getum við notað eina **if..then..endif** setningu í stað tveggja með því að nota nýtt orð, **else** (íslenska: annars).

Ef við endurskrifum forritið með því að nota **else**, getur það litið svona út:

If (Clock.Hour < 18) Then

TextWindow.WriteLine("Góðan daginn heimur")

Else

TextWindow.WriteLine("Gott kvöld heimur")

EndIf

Og þetta forrit gerir nákvæmlega það sama og hitt, sem leiðir okkur að mjög mikilvægu lærdómsatriði í tölvuforritun:

“

Í forritun eru venjulega hægt að gera sama hlutinn á marga mismunandi vegu. Stundum er meira vit í að nota eina aðferð fremur en aðra. Valið er í höndum forritarans. Eftir því sem þú skrifar fleiri forrit og öðlast meiri reynslu munt þú byrja að taka eftir mismunandi aðferðum og kostum þeirra og göllum.

# Inndráttur

Í öllum dæmunum getur þú séð að skipanirnar á milli *If*, *Else* og *EndIf* eru dregnar inn frá vinstri. Þessi inndráttur er ekki nauðsynlegur fyrir tölvuna. Hún mundi skilja forritið jafn vel þó honum væri sleppt. En inndrátturinn auðveldar okkur fólkinu að sjá og skilja uppbyggingu forritsins. Þessa vegna eru það talin vera góð vinnubrögð að nota inndrátt á skipanir sem á einhvern hátt flokkast saman.

# Jöfn eða odda

Nú þegar við höfum bætt **If..Then..Else..EndIf** setningunni í verkfærakassann okkar skulum við skrifa forrit sem getur sagt okkur hvort tala er jöfn tala (t.d. 2, 4, 6, …) eða oddatala (t.d. 1, 3, 5,…).

TextWindow.Write("Sláðu inn tölu: ")

talan = TextWindow.ReadNumber()

afgangur = Math.Remainder(talan, 2)

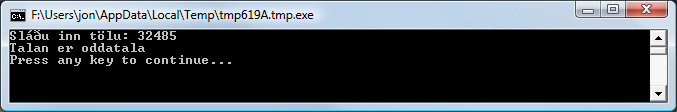
If (afgangur = 0) Then

TextWindow.WriteLine("Talan er jöfn tala")

Else

TextWindow.WriteLine("Talan er oddatala")

EndIf



Mynd 16 – Forritið Jöfn eða odda

Í þessu forriti kynntum við enn eina gagnlega aðgerð, **Math.Remainder**. Og þú hefur ef til vill þegar áttað þig á því að hún deilir fyrri tölunni í sviganum með þeirri síðari (2) og skilar afganginum af deilingunni, sem er 0 ef talan er jöfn tala, en annars 1.

# Greinar

Í öðrum kafla lærðir þú að tölvan les forrit eina línu í einu, frá fyrstu línu til þeirrar síðustu. En það er til sérstök skipun sem getur látið tölvuna stökkva á tiltekna línu í forriti og halda áfram að lesa þaðan. Við skulum skoða þetta í næsta forriti.

i = 1

byrjun:

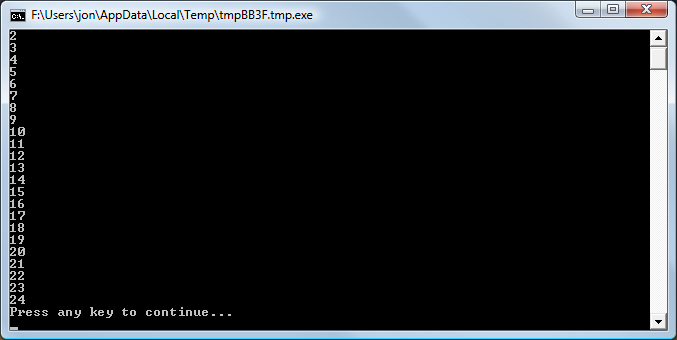
TextWindow.WriteLine(i)

i = i + 1

If (i < 25) Then

Goto byrjun

EndIf



Mynd 17 – Notkun GoTo

Í forritinu hér fyrir ofan settum við gildið 1 í breytuna **i**. Síðan settum við nýja setningu sem endar á tvípunkti ( : )

byrjun:

Þetta er kallað *merki* (enska: label). Merki eru eins og bókmerki sem tölvan skilur. Þú getur gefið merkjum hvaða heiti sem þú vilt og haft mörg í sama forriti, en engin tvö mega þó hafa sama heiti.

Önnur athyglisverð skipun hér er:

i = i + 1

Hér er 1 bætt við innihaldið í breytunni **i** og útkoman úr því er svo aftur sett í **i**. Þannig að ef innihald **i** var 1 áður en skipunin var framkvæmd, verður innihaldið 2 á eftir.

Og að lokum

If (i < 25) Then

Goto byrjun

EndIf

Þessi hluti segir tölvunni að ef innihald **i** er minna en 25 þá eigi tölvan að stökkva yfir á merkið **byrjun** og hefja lesturinn þaðan (sem hér hefur þau áhrif að hún endurtekur nokkrar línur).

# Endalaus keyrsla

Með því að nota **Goto** setningu getum við látið tölvuna endurtaka eitthvað eins oft og við viljum. Þú gætir til dæmis tekið forritið sem finnur út jafnar og oddatölur og breytt því eins og sýnt er hér fyrir neðan og forritið mun keyra endalaust. Þú getur stöðvað forritið með því að smella á Loka (X) hnappinn á horni gluggans efst hægra megin.

byrjun:

TextWindow.Write("Sláðu inn tölu: ")

talan = TextWindow.ReadNumber()

afgangur = Math.Remainder(talan, 2)

If (afgangur = 0) Then

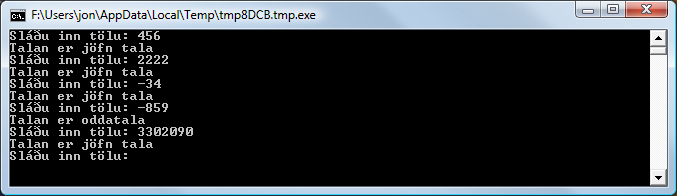
TextWindow.WriteLine("Talan er jöfn tala")

Else

TextWindow.WriteLine("Talan er oddatala")

EndIf

Goto byrjun



Mynd 18 – Forritið Jöfn eða odda í endalausri keyrslu

Kafli 5

Lykkjur

# For lykkjan

Við skulum skoða aftur forritið sem við skrifuðum í kaflanum hér á undan.

i = 1

byrjun:

TextWindow.WriteLine(i)

i = i + 1

If (i < 25) Then

Goto byrjun

EndIf

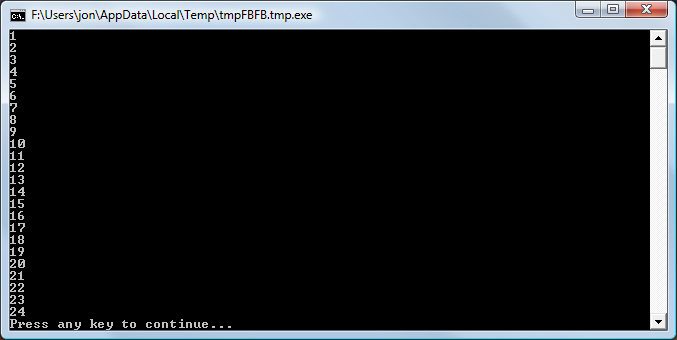
Þetta forrit skrifar út tölur í röð frá 1 til 24. Það er svo algengt í forritun að gera svona endurteknar breytingar á innihaldi breytu að forritunarmál eru venjulega með sérstakar og auðveldari aðferðir til þess. Forritið hér fyrir neðan gerir það sama og forritið fyrir ofan:

For i = 1 To 24

TextWindow.WriteLine(i)

EndFor

Og útkoman er:



Mynd 19 - For lykkjan notuð

Taktu eftir að við gátum gert að sama í 3 línum og við gerðum áður í 7 línum! Mundu hvað við sögðum fyrr um að það væri yfirleitt hægt að framkvæma sama hlutinn á mismunandi vegu? Þetta er gott dæmi.

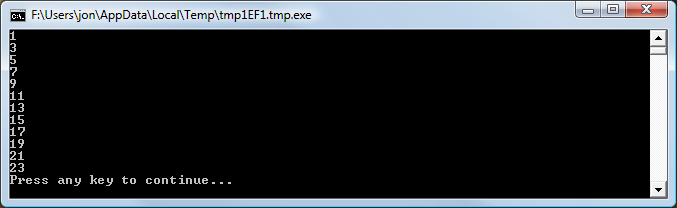
**For..EndFor** er það sem á tungumáli forritunar er kalla *lykkja* (enska: loop). Með þessari lykkju getum við tekið breytu, gefið henni upphaflegt innihald, ákveðið hvaða lokagildi breytan á að hafa og síðan látið tölvuna um að breyta innihaldi hennar í þrepum þar til hún hefur fengið lokagildið. Í hvert sinn sem tölvan breytir innihaldi breytunnar keyrir hún aftur skipanirnar á milli **For** og **EndFor**.

Í þessu forriti bættist 1 við breytuna í hverri umferð lykkjunnar. En það væri hægt að nota einhverja aðra tölu t.d. 2 ef þú vildir aðeins skrifa út oddatölur (aðra hverja tölu) frá 1 til 24.

For i = 1 To 24 Step 2

TextWindow.WriteLine(i)

EndFor



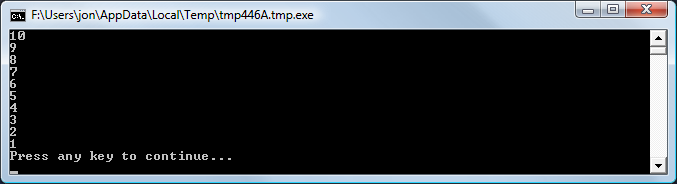
Mynd 20 – Aðeins oddatölurnar

**Step 2** hluti **For** setningarinnar segir tölvunni að hækka innihald **i** um 2 í staðinn fyrir 1 sem er notað ef Step er ekki tiltekið. Með **Step** er hægt að tiltaka hvaða breytingu sem er. Það er meira að segja hægt að nota neikvæða tölu og láta tölvuna telja aftur á bak, eins og þetta dæmi sýnir:

For i = 10 To 1 Step -1

TextWindow.WriteLine(i)

EndFor



Mynd 21 – Talið aftur á bak

# While lykkjan

**While** lykkjan er önnur lykkjuaðferð, sem er sérstaklega gagnleg þegar það er ekki vitað fyrirfram hve margar umferðir lykkjan á að fara. Ólíkt *For* lykkjunni þar sem fjöldi umferða er þekktur fyrirfram, er *While* lykkjan endurtekin á meðan tiltekið skilyrði er uppfyllt (enska orðið *while* þýðir jú *á meðan*). Í dæminu fyrir neðan er tölu skipt í tvennt í hverri umferð en aðeins á meðan hún er stærri en 1.

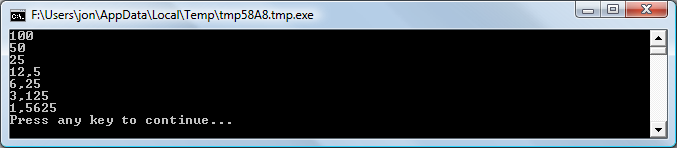
talan = 100

While (talan > 1)

TextWindow.WriteLine(talan)

talan = talan / 2

EndWhile



Mynd 22 – Lykkja sem helmingar

Í forritinu hér fyrir ofan setjum við gildið 100 í breytuna *talan* og keyrum svo *while* lykkjuna á meðan innihald breytunnar *talan* er stærri en 1. Innan lykkjunnar skrifum við út töluna eins og hún er í hverri umferð og deilum henni síðan með 2, sem skiptir henni til helminga. Og eins og búast mátti við er útkoman röð af tölum þar sem hver tala er helmingurinn af næstu tölu á undan.

Það yrði mjög erfitt að skrifa þetta forrit með því að nota *For* lykkju, vegna þess að við vitum ekki hve oft lykkjan þarf að keyra. Með *while* lykkju er auðvelt að athuga hvort skilyrði er uppfyllt og segja tölvunni annaðhvort að endurtaka lykkjuna eða fara út úr henni.

Það er vert að taka eftir því að öllum *while* lykkjum er hægt að breyta í setningar sem nota *If..Then..EndIf*. Til dæmis er hægt að umskrifa forritið hér fyrir ofan eins og hér er sýnt án þess að það breyti útkomunni.

talan = 100

byrjun:

TextWindow.WriteLine(talan)

talan = talan / 2

If (talan > 1) Then

Goto byrjun

EndIf

*Reyndar er það svo að tölvan túlkar ekki While lykkjur beint heldur umbreytir þeim fyrst innra með sér í skipanir sem nota If..Then ásamt einni eða fleiri Goto skipunum sem tölvan framkvæmir síðan.*

Kafli 6

Byrjað í myndrænu umhverfi

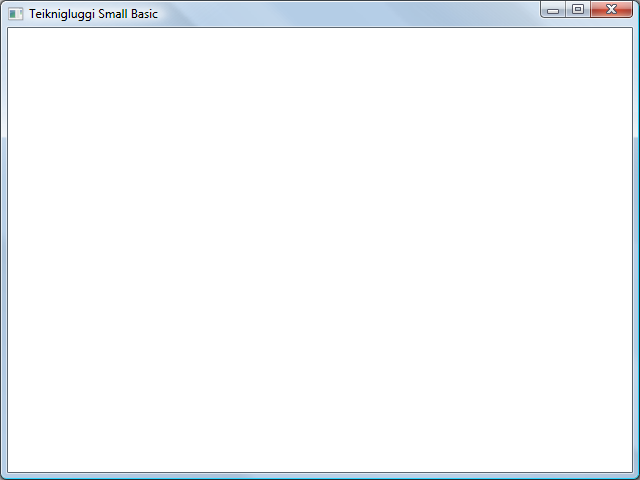
Hingað til höfum við í öllum dæmum okkar notað textagluggann (TextWindow hlutinn) til að útskýra grunnatriði Small Basic forritunarmálsins. En Small Basic hefur öflugar aðferðir til að vinna með í myndrænu umhverfi og í þessum kafla er ætlunin að byrja að skoða þær.

# Kynning á teikniglugganum

Á sama hátt og við gátum notað textagluggann (gegnum TextWindow hlutinn) til að vinna með texta og tölur er Small Basic með svonefndan **teikniglugga** (**GraphicsWindow** hlut) þar sem við getum sett gögn fram á myndrænu formi. Við skulum byrja á skipun til að gera teiknigluggann sýnilegan.

GraphicsWindow.Show()

Þegar þetta forrit er keyrt fáum við ekki venjulega svarta textagluggann heldur hvítan glugga eins og þann sem myndin sýnir. Þetta er venjulegur Windows gluggi en alveg tómur til að byrja með. Í þessum og síðari köflum munum við nota þennan glugga fyrir flest sem við gerum. Þú getur lokað glugganum með því að smella á X hnappinn efst til hægri á glugganum.



Mynd 23 – Tómur teiknigluggi

## Uppstilling teiknigluggans

Þú getur sérsniðið útlit teiknigluggans eins og þú vilt. Það er til dæmis hægt að breyta titlinum, bakgrunnslitnum og stærðinni. Við skulum prófa að gera nokkrar breytingar í þeim tilgangi að venjast þessum glugga.

GraphicsWindow.BackgroundColor = "SteelBlue"

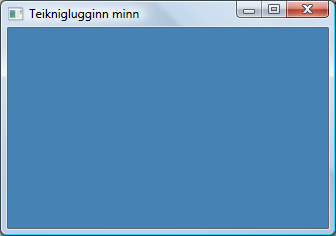
GraphicsWindow.Title = "Teikniglugginn minn"

GraphicsWindow.Width = 320

GraphicsWindow.Height = 200

GraphicsWindow.Show()

Sérsniðni teikniglugginn ætti að líta svona út. Það er hægt að breyta bakgrunnslitnum með því að nota eitthvað af gildunum sem eru talin upp í Viðauka B. Gerðu tilraunir með þessa eiginleika til að komast að því hvernig þú getur breytt útliti teiknigluggans.



Mynd 24 – Sérsniðinn teiknigluggi

# Að teikna línur

Innan teiknigluggans getum við teiknað form, texta og jafnvel myndir. Við skulum byrja á nokkrum einföldum formum. Hér er forrit sem teiknar tvær línur í teikniglugganum.

GraphicsWindow.Width = 200

GraphicsWindow.Height = 200

GraphicsWindow.DrawLine(10, 10, 100, 100)

GraphicsWindow.DrawLine(10, 100, 100, 10)



Mynd 25 – Línur í kross

Fyrstu tvær línur forritsins stilla upp glugganum og næstu tvær teikna línur í kross. DrawLine hefur fjórar færibreytur. Fyrstu tvær eru tölur sem eru x- og y-hnit byrjunarpunkts línunnar sem á að teikna. Hinar tvær tölurnar segja til um x- og y-hnit endapunktsins. Athyglisvert við þessa tölvuteiknun er að hnitin (0,0) byrja efst í vinstra horni teiknigluggans.



Mynd 26 – Hnit eru talin frá efra vinstra horni teiknigluggans

Small Basic hefur ýmsar aðferðir til að breyta eiginleikum línu, svo sem lit hennar og breidd. Við skulum byrja á að breyta lit línanna í forritinu eins og sýnt er hér fyrir neðan.

GraphicsWindow.Width = 200

GraphicsWindow.Height = 200

**GraphicsWindow.PenColor = "Green"**

GraphicsWindow.DrawLine(10, 10, 100, 100)

**GraphicsWindow.PenColor = "Gold"**

GraphicsWindow.DrawLine(10, 100, 100, 10)



*Í stað þess að nota heiti lita getur þú notað RGB ritháttinn fyrir liti á vefnum (#RRGGBB). Til dæmis má nota #FF0000 í staðinn fyrir Red (rautt), #FFFF00 í staðinn fyrir Yellow (gult) o.s.frv. RGB gildi fyrir litaheitin eru gefin í Viðauka B.*

Mynd 27 – Línulitnum beytt

Nú skulum við líka breyta breidd línanna. Í forritinu fyrir neðan stillum við línubreiddina á 10 í stað 1, sem er sjálfgefna breiddin.

GraphicsWindow.Width = 200

GraphicsWindow.Height = 200

**GraphicsWindow.PenWidth = 10**

GraphicsWindow.PenColor = "Green"

GraphicsWindow.DrawLine(10, 10, 100, 100)

GraphicsWindow.PenColor = "Gold"

GraphicsWindow.DrawLine(10, 100, 100, 10)



Mynd 28 – Breiðar línur í lit

Með *PenWidth* og *PenColor* eiginleikunum getum við breytt pennanum sem er notaður til að teikna þessar línur. Þessir eiginleikar hafa ekki aðeins áhrif á línur heldur á öll form sem eru teiknuð eftir að eiginleikunum hefur verið breytt.

Við lærðum í fyrri köflum að nota lykkjuskipanir svo nú getum við auðveldlega skrifað forrit sem teiknar endurteknar línur með vaxandi breidd.

GraphicsWindow.BackgroundColor = "Black"

GraphicsWindow.Width = 200

GraphicsWindow.Height = 160

GraphicsWindow.PenColor = "Blue"

For i = 1 To 10

GraphicsWindow.PenWidth = i

GraphicsWindow.DrawLine(20, i \* 15, 180, i \* 15)

endfor



Mynd 29 – Mismunandi línubreiddir

Athyglisverði hluti þessa forrits er lykkjan, þar sem við hækkum gildi *PenWidth* í hverri umferð lykkjunnar og teiknum svo nýja línu neðan við fyrri línu.

# Að teikna og fylla form

Fyrir hvert form sem hægt er að teikna er venjulega hægt að velja um tvær aðferðir. Það eru annars vegar *Draw* aðgerðir og hins vegar *Fill* aðgerðir. *Draw* aðgerðir teikna útlínu forms með penna (pen) og *Fill* aðgerðir mála fyllt form með því að nota pensil (brush). Sem dæmi eru í forritinu hér fyrir neðan teiknaðir tveir rétthyrningar. Annar er teiknaður með rauðum penna en hinn er teiknaður (og fylltur) með grænum pensli.

GraphicsWindow.Width = 400

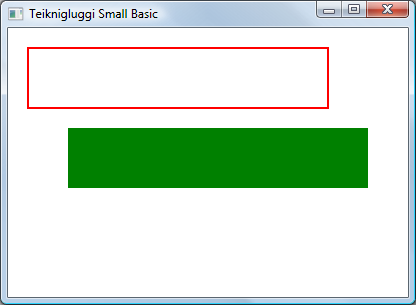
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"

GraphicsWindow.DrawRectangle(20, 20, 300, 60)

GraphicsWindow.BrushColor = "Green"

GraphicsWindow.FillRectangle(60, 100, 300, 60)



Mynd 30 Munurinn á Draw og Fill aðgerðum

Til að teikna rétthyrning þarf fjórar tölur. Fyrstu tvær tölurnar segja til um x- og y-hnit þess horns á rétthyrningum sem er efst til vinstri. Þriðja talan segir til um breidd rétthyrningsins og sú fjórða um hæð hans. Það sama gildir um það að teikna bauga (t.d. sporbauga eða hringi) eins og forritið hér fyrir neðan sýnir.

GraphicsWindow.Width = 400

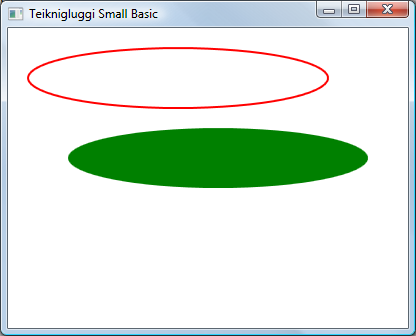
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"

GraphicsWindow.DrawEllipse(20, 20, 300, 60)

GraphicsWindow.BrushColor = "Green"

GraphicsWindow.FillEllipse(60, 100, 300, 60)



Mynd 31 – Að teikna og fylla bauga

Hringar eru sérstakt tilfelli af baugum (enska: ellipses). Til að fá hring verða breidd og hæð baugs að vera jöfn.

GraphicsWindow.Width = 400

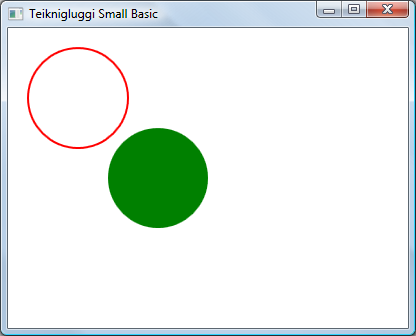
GraphicsWindow.Height = 300

GraphicsWindow.PenColor = "Red"

GraphicsWindow.DrawEllipse(20, 20, 100, 100)

GraphicsWindow.BrushColor = "Green"

GraphicsWindow.FillEllipse(100, 100, 100, 100)



Mynd 32 – Hringir

Kafli 7

Leikið með form

Í þessum kafla ætlum við að leika okkur með það sem hefur verið farið í fram að þessu. Í kaflanum eru nokkur dæmi um áhugaverð forrit sem sameina flest af því sem þú hefur lært.

# Ferningafjöld

Hér teiknum við marga rétthyrninga í lykkju og stækkum þá í hverri umferð.

GraphicsWindow.BackgroundColor = "Black"

GraphicsWindow.PenColor = "LightBlue"

GraphicsWindow.Width = 200

GraphicsWindow.Height = 200

For i = 1 To 100 Step 5

GraphicsWindow.DrawRectangle(100 - i, 100 - i, i \* 2, i \* 2)

EndFor



Mynd 33 - Ferningafjöld

# Hringagnótt

Afbrigði af forritinu á undan sem teiknar hringa í stað ferninga.

GraphicsWindow.BackgroundColor = "Black"

GraphicsWindow.PenColor = "LightGreen"

GraphicsWindow.Width = 200

GraphicsWindow.Height = 200

For i = 1 To 100 Step 5

GraphicsWindow.DrawEllipse(100 - i, 100 - i, i \* 2, i \* 2)

EndFor



Mynd 34 – Hringagnótt

# Litahappdrætti

Þetta forrit notar aðgerðina *GraphicsWindow.GetRandomColor* til að velja liti af handahófi til að setja í pensilinn og notar síðan *Math.GetRandomNumber* til að velja tölur af handahófi sem eru svo settar í x- og y-hnit fyrir hringana (doppurnar). Þessar tvær aðgerðir er hægt að nota saman til að búa til eftirtektarverð forrit sem gefa mismunandi útkomur í hvert sinn sem þau eru keyrð.

GraphicsWindow.BackgroundColor = "Black"

For i = 1 To 1000

GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()

x = Math.GetRandomNumber(640)

y = Math.GetRandomNumber(480)

GraphicsWindow.FillEllipse(x, y, 10, 10)

EndFor



Mynd 35 – Litahappdrætti

# Brotamynd

Eftirfarandi forrit teiknar einfalda brotamynd (enska: fractal) úr þríhyrningi með því að nota handahófstölur. Brotamynd er mynstur sem samanstendur af smærri einingum sem hver um sig er nákvæmlega eins og mynstrið sem þær mynda. Í þessu tilfelli teiknar forritið hundruði þríhyrninga sem hver um sig er eins og stærri þríhyrningur sem hann er hluti af. Og þar sem það tekur forritið nokkurn tíma að keyra er hægt að sjá hvernig þríhyrningarnir eru myndaðir úr dílum á skjánum. Það er svolítið erfitt að útskýra hugsunina í forritinu og það verður ekki gert hér en það er um að gera fyrir þig að kanna það nánar.

GraphicsWindow.BackgroundColor = "Black"

x = 100

y = 100

For i = 1 To 100000

r = Math.GetRandomNumber(3)

ux = 150

uy = 30

If (r = 1) then

ux = 30

uy = 1000

EndIf

If (r = 2) Then

ux = 1000

uy = 1000

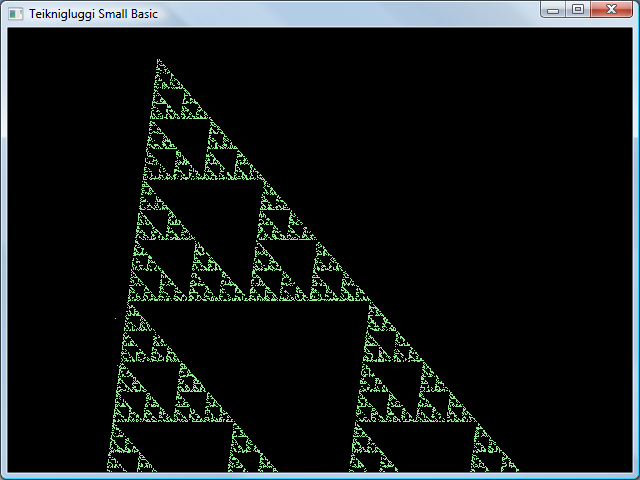
EndIf

x = (x + ux) / 2

y = (y + uy) / 2

GraphicsWindow.SetPixel(x, y, "LightGreen")

EndFor



Mynd 36 – Brotamynd úr þríhyrningum

Ef þú vilt betur sjá dílana mynda mynstrið, getur þú sett töf inn í lykkjuna með því að nota **Program**.*Delay* aðgerðina. Þessi aðgerð tekur inn tölu sem segir til hve löng töfin á að vera í millisekúndum. Hér er forritið aftur með breytingunni sem er auðkennd með feitletrun.

GraphicsWindow.BackgroundColor = "Black"

x = 100

y = 100

For i = 1 To 100000

r = Math.GetRandomNumber(3)

ux = 150

uy = 30

If (r = 1) then

ux = 30

uy = 1000

EndIf

If (r = 2) Then

ux = 1000

uy = 1000

EndIf

x = (x + ux) / 2

y = (y + uy) / 2

GraphicsWindow.SetPixel(x, y, "LightGreen")

**Program.Delay(2)**

EndFor

Aukin töf gerir forritið hægvirkara. Gerðu tilraunir með mismunandi tölur til að finna út hvað hentar þér best.

Önnur breyting sem þú getur gert á forritinu er að skipta út þessari línu:

GraphicsWindow.SetPixel(x, y, "LightGreen")

fyrir þessar tvær:

litur = GraphicsWindow.GetRandomColor()

GraphicsWindow.SetPixel(x, y, litur)

Þessi breyting verður til að þess að forritið teiknar dílana sem mynda þríhyrningana í litum sem eru valdir af handahófi.

Kafli 8

Trítluteiknun

# Logo

Upp úr 1970 kom fram mjög einfalt en jafnframt öflugt forritunarmál nefnt Logo sem var notað af nokkrum vísindamönnum. Það er að segja þangað til einhverjum datt í hug að bæta við því sem nefndist "Turtle Graphics" eða skjaldbökuteiknun, sem notaði "skjaldböku" sem var sýnileg á skjá og gat hlýtt skipunum eins og *Fara áfram*, *Snúa til hægri*, *Snúa til vinstri* o.s.frv. Með skjaldbökunni gat fólk teiknað athyglisverð mynstur á skjáinn. Þetta gerði forritunarmálið aðgengilegt og aðlaðandi fyrir fólk á öllum aldri og leiddi til þess að það varð gífurlega vinsælt upp úr 1980.

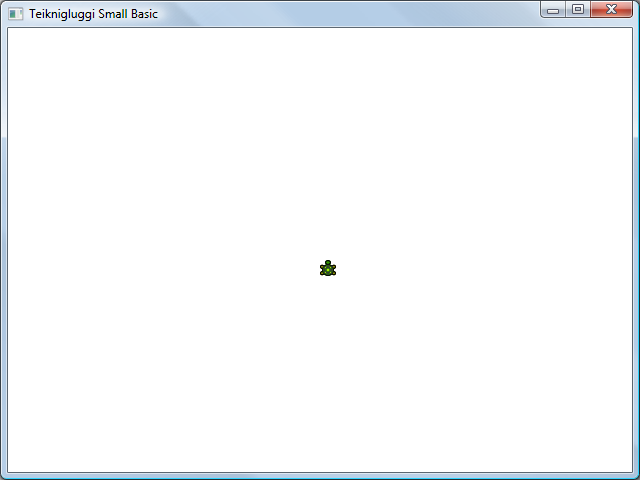
Small Basic hefur skjaldböku og **Turtle** hlut með mörgum skipunum sem hægt er að nota innan úr í Small Basic forritum til að stjórna skjaldbökunni. Í þessum kafla munum við nota skjaldbökuna til að teikna form á skjáinn. *(Athugasemd þýðanda: Þar sem skjaldbaka er fremur óþjált orð, hef ég kosið að gefa skjaldbökunni nafnið Trítla og mun nota það í kaflanum. Hún heitir það einnig í íslenskri úgáfu forritsins.)*

# Skjaldbakan Trítla

Til að byrja þurfum við að gera Trítlu sýnilega á skjánum. Þetta er hægt að gera með einföldu einnar línu forriti.

Turtle.Show()

Þegar þetta forrit er keyrt opnast hvítur teiknigluggi eins og við sáum í fyrri kafla og skjaldbakan situr í honum miðjum og horfir upp. Þetta er hún Trítla sem ætlar að fylgja fyrirmælum okkar og gera hvaðeina sem við biðjum hana um.



Mynd 37 – Skjaldbakan Trítla er sýnileg

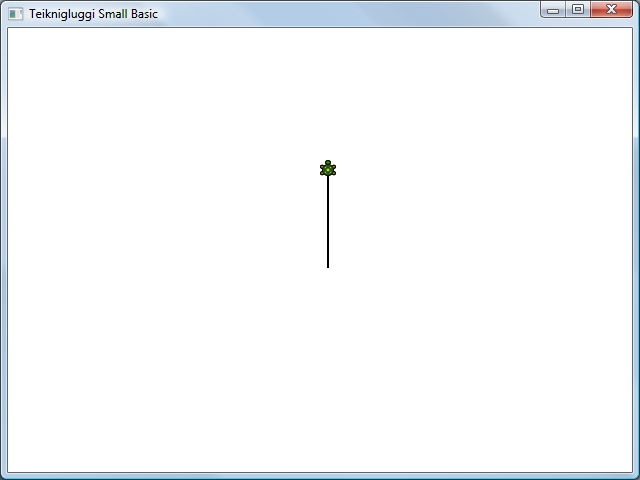
# Hreyfing og teiknun

Ein af skipununum sem Trítla skilur er **Move** (færa). Þessi aðgerð tekur tölu sem færibreytu. Talan segir Trítlu hve langt hún á að færast. Í dæminu hér fyrir neðan ætlum við að biðja Trítlu að færa sig um 100 díla.

Turtle.Move(100)

Þegar þú keyrir þetta forrit sérðu Trítlu færa sig hægt upp um 100 díla á skjánum. Það má líka sjá að þegar hún hreyfir sig skilur hún eftir sig teiknaða línu. Þegar Trítla hefur lokið við að hreyfa sig verður útkoman eitthvað svipuð og sýnt er í myndinni hér fyrir neðan.

*Þegar Turtle aðgerðir eru notaðar er ekki nauðsynlegt að kalla fyrst á Show(). Trítla birtist sjálfkrafa þegar einhver aðgerð Turtle hlutarins er notuð.*



Mynd 38 – Trítla færð um 100 díla

# Að teikna ferning

Ferningur er með fjórar hliðar, tvær lóðréttar og tvær láréttar. Til að teikna ferning getum við látið Trítlu fyrst teikna línu og síðan snúa til hægri. Ef við látum hana svo endurtaka þessar tvær aðgerðir þrisvar sinnum í viðbót þá fáum við ferning. Í forriti gæti þetta litið svona út.

Turtle.Move(100)

Turtle.TurnRight()

Turtle.Move(100)

Turtle.TurnRight()

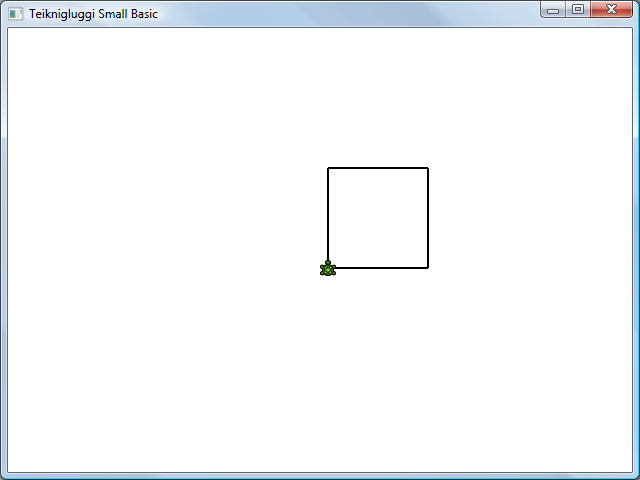
Turtle.Move(100)

Turtle.TurnRight()

Turtle.Move(100)

Turtle.TurnRight()

Þegar forritið er keyrt sjáum við Trítlu teikna línu og snúa til hægri fjórum sinnum til að mynda ferning:



Mynd 39 – Trítla að teikna ferning

Það er athyglisvert að taka eftir því að við erum að gefa sömu tvær skipanirnar aftur og aftur – eða fjórum sinnum í allt. Og við höfum nú þegar lært að svona endurteknar skipanir er hægt að framkvæma með því að nota lykkjur. Svo ef við tökum forritið hér á undan og breytum því með því að nota **For..EndFor** lykkju getum við notast við færri línur.

For i = 1 To 4

Turtle.Move(100)

Turtle.TurnRight()

EndFor

# Skipt um liti

Trítla teiknar í sama teikniglugganum sem við kynntumst í kaflanum á undan. Það þýðir að við getum notað hér allar aðgerðir sem við lærðum í þeim kafla. Til dæmis getum við teiknað hverja hlið ferningsins í nýjum lit eins og þetta forrit sýnir.

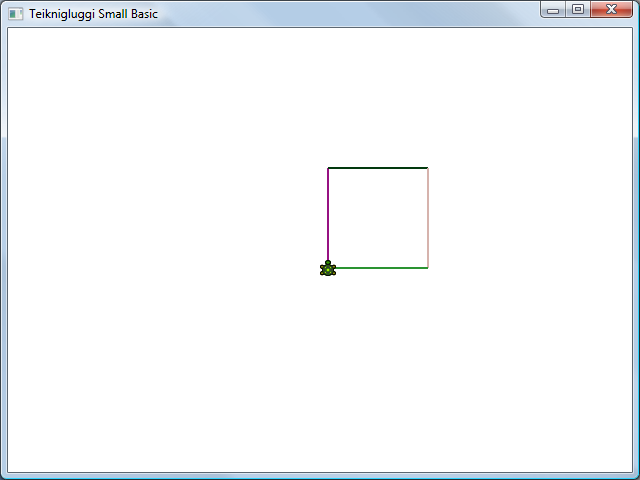
For i = 1 To 4

GraphicsWindow.PenColor = GraphicsWindow.GetRandomColor()

Turtle.Move(100)

Turtle.TurnRight()

EndFor



Mynd 40 – Skipt um liti

# Teiknun flóknari forma

Auk aðgerðanna **TurnRight** og **TurnLeft** sem beygja um rétt 90 gráðu horn hefur *Turtle* hluturinn aðgerðina **Turn** en með henni getum við látið Trítlu snúa um hvaða horn sem er. Aðgerðin tekur inn eina tölu (gráðurnar) og með því að nýta hana er hægt að teikna alls konar form með mismunandi hliðar og horn. Eftirfarandi forrit teiknar sexhyrning (marghyrning með sex hliðar).

For i = 1 To 6

Turtle.Move(100)

Turtle.Turn(60)

EndFor

Prófaðu þetta forrit til að komast að því hvort það teiknar sexhyrning í raun og veru. Taktu eftir að þar sem hornið á milli samliggjandi hliða sexhyrnings er 60 gráður notum við **Turn(60)**. Fyrir svona reglulegan marghyrning er auðvelt að finna út hornið milli hliða með því að deila 360 með fjölda hliðanna. Með þessar upplýsingar og með því að nota breytur getum við skrifað fremur almennt forrit sem getur teiknað hvaða marghyrning sem er.

hliðarAlls = 12

hlið = 400 / hliðarAlls

horn = 360 / hliðarAlls

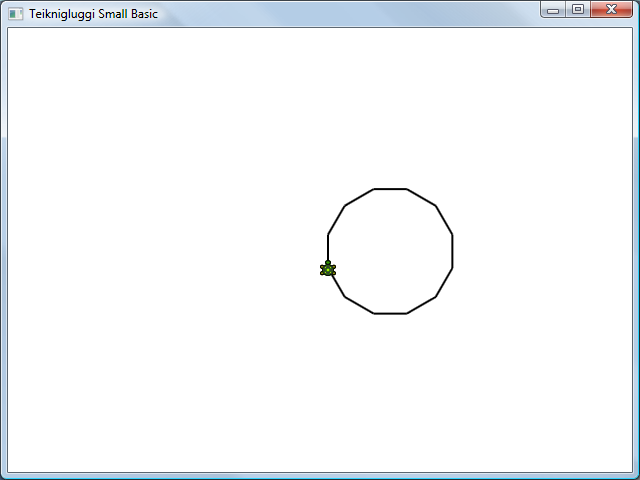
For i = 1 To hliðarAlls

Turtle.Move(hlið)

Turtle.Turn(horn)

EndFor

Með þessu forriti getur þú teiknað hvaða marghyrning sem með því einu að setja annað gildi í breytuna **hliðarAlls**. Talan 4 mundi gefa okkur ferninginn sem við byrjuðum með. Ef nógu stór tala er sett, t.d. 50 verður sýnilega útkoman sú sama og að teikna hring.



Mynd 41 – Tólf hliða marghyrningur teiknaður

Með aðferðinni sem við vorum að læra getum við nú fengið athyglisverða útkomu með því að láta Trítlu teikna marga hringi og bætt inn smá snúningi eftir hvern.

hliðarAlls = 50

hlið = 400 / hliðarAlls

horn = 360 / hliðarAlls

Turtle.Speed = 9

*Í þessu forriti höfum við látið Trítlu fara hraðar með því að stilla Speed á 9. Þú getur stillt þennan eiginleika á tölu frá 1 til 10 til að stjórna hve hratt þú vilt láta hana trítla.*

For j = 1 To 20

For i = 1 To hliðarAlls

Turtle.Move(hlið)

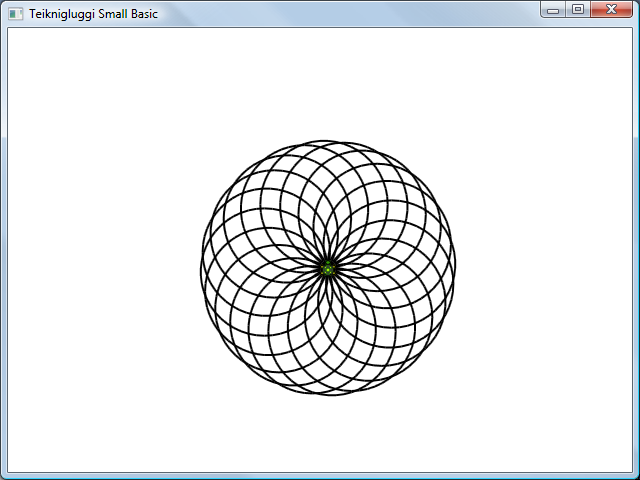
Turtle.Turn(horn)

EndFor

Turtle.Turn(18)

EndFor

Forritið hér á undan hefur tvær **For..EndFor** lykkjur þar sem önnur lykkjan er innan í hinni. Innri lykkjan (i = 1 til hliðarAlls) er svipuð og forritið sem teiknaði marghyrningana. Ytri lykkjan (j = 1 to 20) sér um að snúa Trítlu aðeins eftir hvern hring sem er teiknaður og að teikna 20 hringi (endurtaka innri lykkjuna 20 sinnum). Þegar lykkjurnar eru notaðar svona saman fáum við athyglisvert mynstur eins og sýnt er hér fyrir neðan.



Mynd 42 - Hringsnúningur

# Hreyfingar

Þú getur látið Trítlu *ekki* teikna með **PenUp** aðgerðinni sem lyftir teiknipennanum. Þannig getur þú fært Trítlu hvert sem er á skjánum án þess að teikna línu um leið. Aðgerðin **PenDown** setur pennann niður þannig að Trítla teiknar á ný. Þetta er hægt að nota á ýmsan hátt, t.d. til að teikna brotalínu. Hér er forrit sem notar þetta til að teikna marghyrning með brotinni línu.

hliðarAlls = 6

hlið = 400 / hliðarAlls

horn = 360 / hliðarAlls

For i = 1 To hliðarAlls

For j = 1 To 6

Turtle.Move(hlið / 12)

Turtle.PenUp()

Turtle.Move(hlið / 12)

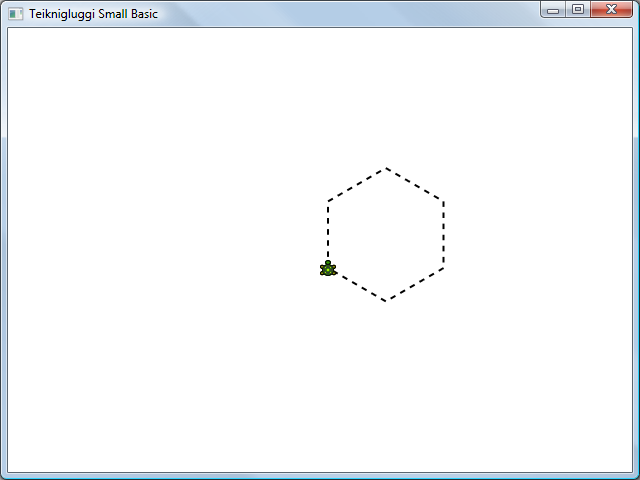
Turtle.PenDown()

EndFor

Turtle.Turn(horn)

EndFor

Eins og forritið áður hefur þetta tvær lykkjur. Innri lykkjan teiknar eina hlið með brotinni línu en ytri lykkjan segir til um hve margar línur á að teikna. Í dæminu settum við **6** í breytuna **hliðarAlls** og fengum þess vegna sexhyrning teiknaðan með brotalínum eins og sést hér fyrir neðan.



Mynd 43 – Notkun PenUp og PenDown

Kafli 9

Stefjur

Mjög oft þegar við skrifum forrit lendum við í tilfellum þar sem við þurfum að láta forrit framkvæma sömu röð skipana aftur og aftur. Í svoleiðis tilfellum viljum við yfirleitt losna við að skrifa sömu skipanir aftur og aftur á mismunandi stöðum í forritinu. Það er við svona aðstæður sem *stefjur* (enska: subroutines) koma að góðum notum.

Stefja er hluti af stærra forriti og hefur venjulega einhvern sérstakan tilgang. Stefjuna er hægt að kalla á hvar sem er í forritinu líkt og um nýja skipun væri að ræða. Upphaf stefju er auðkennt með **Sub** lykilorðinu ásamt heiti stefjunnar og **EndSub** lykilorðið markar enda stefjunnar. Eftirfarandi bútur af forritskóða sýnir stefju sem heitir *SkrifaTímann* og gerir ekki annað en skrifa út í textagluggann hvað klukkan er.

Sub **SkrifaTímann**

TextWindow.WriteLine(Clock.Time)

EndSub

Hér fyrir neðan er forrit til að sýna notkun stefju. Það inniheldur stefjuna og kallar á hana tvisvar.

TextWindow.Write("Klukkan er nú: ")

SkrifaTímann()

TextWindow.Write("Skrifaðu nafnið þitt: ")

nafn = TextWindow.Read()

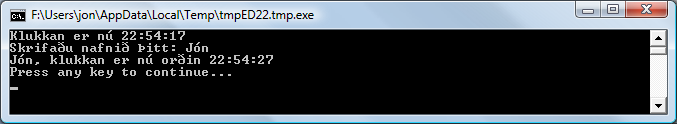
TextWindow.Write(nafn + ", klukkan er nú orðin: ")

SkrifaTímann()

Sub SkrifaTímann

TextWindow.WriteLine(Clock.Time)

EndSub



Mynd 44 – Kallað á einfalda stefju

Stefjan er framkvæmd með því að nota heiti hennar með sviga, *Stefjuheiti( )*. Eins og venjulega eru greinarmerkin, í þessu tilfelli svigarnir, nauðsynlegir til að tölvan skilji hvað á að gera, þ.e. að hún eigi að finna stefjuna og framkvæma skipanir hennar, en halda síðan áfram að lesa aðalforritið.

# Kostir þess að nota stefjur

Eins og við sáum hér fyrir ofan, hjálpa stefjur til að fækka þeim línum af forritskóða sem við þurfum að skrifa. Þegar þú hefur einu sinni leyst það hvernig á að skrifa út tímann getur þú nýtt þessa lausn hvar sem er í forritinu til að skrifa út tímann. *c*

Auk þess hjálpa stefjur okkur að kljúfa flókin úrlausnarefni í einfaldari einingar. Ef þú þyrftir til dæmis að leysa flókna stærðfræðijöfnu, gætir þú skrifað nokkrar stefjur sem hver um sig mundi leysa einhvern hluta af hinni flóknu jöfnu. Þú gætir síðan sett stefjurnar saman í forrit til að leysa upphaflega verkefnið. (Forrit eru oft lítið annað en safn af skipunum sem kalla á alls konar sérhæfðar stefjur.)

*Mundu að þú getur aðeins kallað á Small Basic stefju sem er inni í forriti. Þú getur ekki kallað á stefjur sem eru hluti af öðrum forritum (sem takmarkar notagildið talsvert).*

Stefjur geta einnig gert forrit læsilegri. Með öðrum orðum, ef þú notar stefjur fyrir algengar aðgerðir í forriti og gefur stefjunum lýsandi heiti þá verður auðveldara að lesa og skilja forritið. Þetta er mjög mikilvægt ef þú vilt skilja forrit sem aðrir hafa skrifað eða ef þú vilt að forritið þitt sé auðskilið fyrir aðra. Stundum hjálpar það meira að segja manni sjálfum að skilja eigin forrit, t.d. þegar einhver tími er liðinn frá því forritið var skrifað.

# Notkun breytna

Innan stefju hefur þú aðgang að og getur notað hvaða breytu sem er í forriti. Til dæmis tekur eftirfarandi forrit við tveimur tölum og skrifar út hærri töluna. Taktu eftir því að breytan *hámark* er notuð bæði innan og utan stefjunnar.

TextWindow.Write("Sláðu inn tölu: ")

tala1 = TextWindow.ReadNumber()

TextWindow.Write("Sláðu inn aðra tölu: ")

tala2 = TextWindow.ReadNumber()

FinnaHámark()

TextWindow.WriteLine("Hærri talan er: " + hámark)

Sub FinnaHámark

If (tala1 > tala2) Then

hámark = tala1

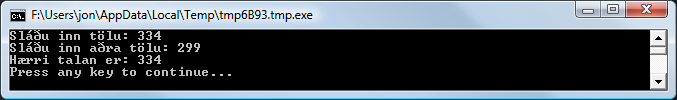
Else

hámark = tala2

EndIf

EndSub

Og útkoman úr forritinu er svipuð þessu:



Mynd 45 – Hærri talan fundin með stefju

Við skulum skoða annað dæmi sem sýnir notkun á stefjum. Í þetta sinn ætlum við að nota teikniforrit sem reiknar út ýmsa punkta sem verða geymdir í breytunum *x* og *y*. Síðan kallar forritið á stefjuna **TeiknaHringUmMiðju** sem sér um að teikna hring með miðju í *x* og *y*.

GraphicsWindow.BackgroundColor = "Black"

GraphicsWindow.PenColor = "LightBlue"

GraphicsWindow.Width = 480

For i = 0 To 6.4 Step 0.17

x = Math.Sin(i) \* 100 + 200

y = Math.Cos(i) \* 100 + 200

TeiknaHringUmMiðju()

EndFor

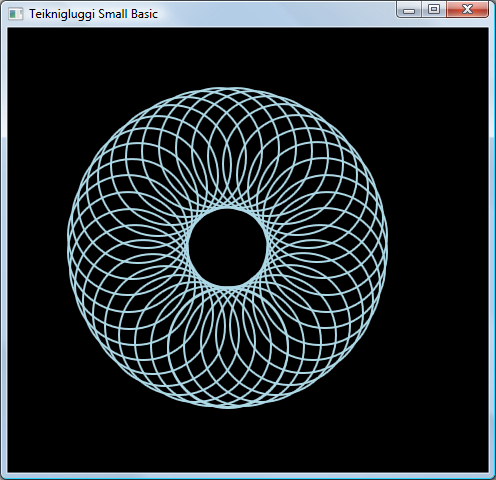
Sub TeiknaHringUmMiðju

fráX = x - 40

fráY = y - 40

GraphicsWindow.DrawEllipse(fráX, fráY, 120, 120)

EndSub



Mynd 46 – Teiknað dæmi með stefju

# Kallað á stefjur innan úr lykkjum

Stundum er kallað á stefjur innan úr lykkjum. Stefjurnar framkvæma sömu skipanir í hvert sinn en með nýjum gildum í einni eða fleiri breytum. Segjum til dæmis að þú hafir stefju sem heitir *PrímtöluAthugun* sem kannar hvort tiltekin tala er prímtala eða ekki. Þú getur skrifað forrit sem biður notandann um tölu og getur þá komist að því með því að nota stefjuna hvort talan er prímtala eða ekki. Forritið hér fyrir neðan sýnir þetta:

TextWindow.Write("Skrifaðu tölu: ")

i = TextWindow.ReadNumber()

erPrím = "True"

PrímtöluAthugun()

If (erPrím = "True") Then

TextWindow.WriteLine(i + " er prímtala")

Else

TextWindow.WriteLine(i + " er ekki prímtala")

EndIf

Sub PrímtöluAthugun

For j = 2 To Math.SquareRoot(i)

If (Math.Remainder(i, j) = 0) Then

erPrím = "False"

Goto EndirLykkju

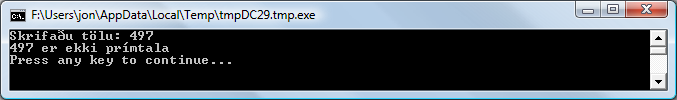
EndIf

Endfor

EndirLykkju:

EndSub

Stefjan *PrímtöluAthugun* tekur gildið *i* og reynir að deila því með minni tölum. Ef *i* er deilt með tölu og afgangurinn er 0 þá er *i* ekki prímtala. Þegar það gerist setur stefjan gildið "False" í breytuna *erPrím* og notar *Goto* grein til að stökkva út úr lykkjunni niður á merkið *Endalykkja*. Ef allar smærri tölur en talan sem er athuguð gefa afgang heldur *erPrím* áfram að hafa gildið "True".



Mynd 47 – Athugun á prímtölum

Nú hefur þú stefju sem getur prófað hvort tala er prímtala og vilt nota hana til að finna allar prímtölur undir til dæmis 100. Það er mjög auðvelt að breyta forritinu hér að ofan og láta það kalla á stefjuna *PrímtöluAthugun* innan úr lykkju. Þetta gefur stefjunni nýtt gildi til að skoða í hverri umferð lykkjunnar. Skoðum hvernig þetta er gert í dæminu hér fyrir neðan.

For i = 3 To 100

erPrím = "True"

PrímtöluAthugun()

If (erPrím = "True") Then

TextWindow.WriteLine(i)

EndIf

EndFor

Sub PrímtöluAthugun

For j = 2 To Math.SquareRoot(i)

If (Math.Remainder(i, j) = 0) Then

erPrím = "False"

Goto EndirLykkju

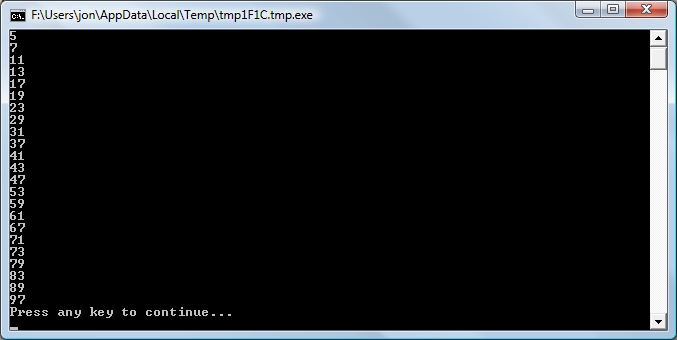
EndIf

Endfor

EndirLykkju:

EndSub

Í forritinu hér fyrir ofan er gildið í *i* uppfært í hverri umferð lykkjunnar. Innan í lykkjunni er kallað á stefjuna *PrímtöluAthugun*. Stefjan tekur töluna í *i* og reiknar út hvort hún er prímtala eða ekki. Útkoman er sett í breytuna *erPrím* sem er síðan notuð í lykkjunni utan við stefjuna. Ef *i* inniheldur prímtölu er innihald þess skrifað út í textagluggann. Og þar sem lykkjan byrjar frá 3 og fer upp í 100 fáum við lista yfir allar prímtölur frá og með 3 til og með 100. Hér fyrir neðan má sjá útkomu úr forritinu.



Mynd 48 - Prímtölur

Kafli 10

Fylki

Nú ættir þú að hafa fengið góða æfingu í að nota breytur – þú ert jú enn hér og hefur ennþá gaman af þessu, ekki satt?

Við skulum nú skoða aftur eitt fyrsta forritið sem við skrifuðum, þar sem við notuðum breytur:

TextWindow.Write("Skrifaðu nafnið þitt: ")

nafn = TextWindow.Read()

TextWindow.WriteLine("Halló " + nafn)

Í þessu forriti tókum við á móti og geymdum nafn notandans í breytu með heitinu **nafn**. Síðar í forritinu sögðum við "Halló" við notandann. Gerum nú ráð fyrir að það sé fleiri en einn notandi – segjum 5 notendur. Hvernig væri hægt að geyma nöfn þeirra allra? Ein aðferð væri þessi:

TextWindow.Write("Notandi 1, skrifaðu nafnið þitt: ")

nafn1 = TextWindow.Read()

TextWindow.Write("Notandi 2, skrifaðu nafnið þitt: ")

nafn2 = TextWindow.Read()

TextWindow.Write("Notandi 3, skrifaðu nafnið þitt: ")

nafn3 = TextWindow.Read()

TextWindow.Write("Notandi 4, skrifaðu nafnið þitt: ")

nafn4 = TextWindow.Read()

TextWindow.Write("Notandi 5, skrifaðu nafnið þitt: ")

nafn5 = TextWindow.Read()

TextWindow.Write("Halló ")

TextWindow.Write(nafn1 + ", ")

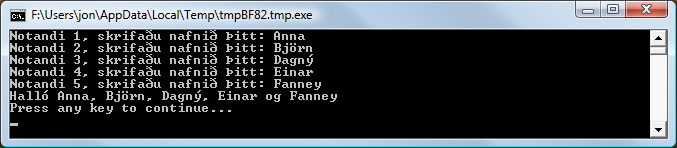
TextWindow.Write(nafn2 + ", ")

TextWindow.Write(nafn3 + ", ")

TextWindow.Write(nafn4 + " og ")

TextWindow.WriteLine(nafn5)

Þegar þetta er keyrt fáum við eftirfarandi útkomu:



Mynd 49 – Lausn án þess að nota fylki

Það hlýtur samt að vera til betri aðferð til að skrifa svona einfalt forrit, eða hvað? Og þar sem tölvan á mjög auðvelt með leysa verkefni með endurtekningu því ættum við þá að þurfa að skrifa sama kóðann aftur og aftur fyrir hvern nýjan notanda? Lausnin hér er að geyma öll nöfnin í einni breytu og sækja þau í hana. Ef við gerum það getum við notað **For** lykkjuna sem við lærðum um í fyrri köflum. Það er hér sem *fylki* koma að góðu gagni.

# Hvað er fylki?

Fylki (enska: array) er sérstök tegund af breytu sem getur innihaldið fleiri en eitt gildi í einu. Í dæminu okkar þýðir það að í stað þess að búa til fimm breytur **nafn1**, **nafn2**, **nafn3**, **nafn4** og **nafn5** til að geyma nöfn fimm notenda, gætum við geymt öll nöfnin í einni breytu **nafn**. Aðferðin til að gera það er að nota fyrirbæri sem nefnist **vísir** (enska: index). Til dæmis geta **nafn[1]**, **nafn[2]**, **nafn[3]**, **nafn[4]** og **nafn[5]** hvert geymt eitt nafn. Tölurnar 1, 2, 3, 4 og 5 eru kallaðar **vísar** fylkisins.

Þó að **nafn[1]**, **nafn[2]**, **nafn[3]**, **nafn[4]** og **nafn[5]** líti út eins og mismunandi breytur eru þær í reynd aðeins ein breyta. Og hvað græðir maður nú eiginlega á þessu? Jú, það besta við að geyma gildi í fylki er að maður getur tiltekið vísinn með því að nota aðra breytu – sem þýðir að það er auðvelt að vinna með breytur innan í lykkjum.

Nú skulum við skoða hvernig við getum notað þessa nýju þekkingu með því að umskrifa forritið frá því áður og nota fylki til að stytta það.

For i = 1 To 5

TextWindow.Write("Notandi " + i + ", skrifaðu nafn þitt: ")

**nafn[i] = TextWindow.Read()**

EndFor

TextWindow.Write("Halló ")

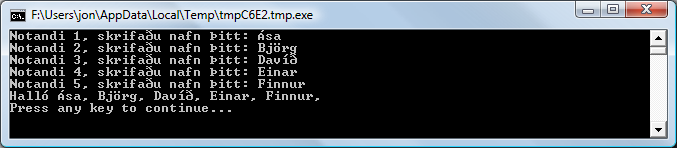
For i = 1 To 5

**TextWindow.Write(nafn[i] + ", ")**

EndFor

TextWindow.WriteLine("")

Er þetta ekki auðveldara að lesa? Taktu eftir feitletruðu línunum tveimur. Sú fyrri setur gildi í fylkið og sú seinni les þau úr fylkinu. Gildið sem er sett í **nafn[2]** hefur engin áhrif á gildið í **nafn[1]**. Þess vegna er hægt að vinna með **nafn[1]** og **nafn[2]** sem tvær ólíkar breytur en þó með sama heiti.



Mynd 50 – Lausn með því að nota fylki

Forritið hér fyrir ofan gefur okkur næstum því sömu útkomu og forritið án fylkis, nema hvað það vantar "og" framan við aftasta nafnið og nafnalistinn endar á aukakommu ",". Við getum leiðrétt það með því að laga lykkjuna sem skrifar út þannig:

TextWindow.Write("Halló ")

For i = 1 To 5

TextWindow.Write(nafn[i])

If i < 4 Then

TextWindow.Write(", ")

EndIf

If i = 4 Then

TextWindow.Write(" og ")

EndIf

EndFor

TextWindow.WriteLine("")

# Vísar í fylki

Í forritinu hér á undan sástu hvernig við gátum notað tölur sem vísa til að setja gildi í fylki og sækja þau aftur úr því. Það vill svo til að vísar takmarkast ekki við tölur og raunar getur verið mjög gagnlegt að nota textavísa líka. Eftirfarandi forrit er dæmi um það en í því biðjum við um og geymum ýmsar upplýsingar um notanda og skrifum síðan út þær upplýsingar sem notandinn biður um.

TextWindow.Writeline("Færðu inn þessar upplýsingar um þig:")

TextWindow.Writeline("------------------------------------")

TextWindow.Write("Nafn: ")

notandi["nafn"] = TextWindow.Read()

TextWindow.Write("Aldur: ")

notandi["aldur"] = TextWindow.Read()

TextWindow.Write("Sveitarfélag: ")

notandi["sveitarfélag"] = TextWindow.Read()

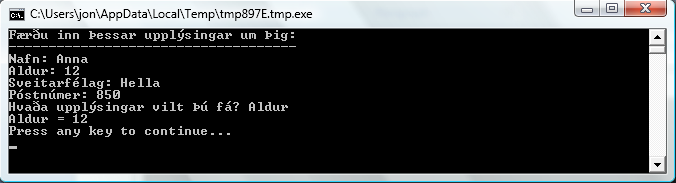
TextWindow.Write("Póstnúmer: ")

notandi["póstnúmer"] = TextWindow.Read()

TextWindow.Write("Hvaða upplýsingar vilt þú fá? ")

vísir = TextWindow.Read()

TextWindow.WriteLine(vísir + " = " + notandi[vísir])



Mynd 51 – Notkun vísa sem ekki eru tölur

# Fleiri en ein vídd

Segjum að þú viljir skrá nöfn og símanúmer allra vina þinna og geta flett upp á þeim þegar þú vilt – líkt og í símaskrá. Hvernig gætum við skrifað slíkt forrit og notað fylki?

Í þetta skipti koma tvenns konar vísar við sögu í sama fylki (hver hópur af vísum í fylki er einnig kallaður *vídd* svo við gætum líka talað um fylkið hafi tvær víddir). Gerum ráð fyrir að við auðkennum vinina með skammstöfun fyrir hvern. Skammstöfun nafns verður þá annar vísirinn (önnur víddin) í fylkinu. Fyrir hinn vísinn (víddina) notum við þá heiti þeirra atriða sem á að skrá upplýsingar um, þ.e. **nafn** og **sími**.

*Þegar unnið er með visa fylkis skipta stórir og litlir stafir ekki máli, ekki frekar en með venjulegar breytur.*

Við mundum skrá gögnin á eftirfarandi hátt (taktu eftir víddunum tveimur í fylkinu *vinir*):

vinir["AJ"]["Nafn"] = "Anna"

vinir["AJ"]["Sími"] = "799-6789"

vinir["BÞ"]["Nafn"] = "Björn"

vinir["BÞ"]["Sími"] = "799-4567"

vinir["SR"]["Nafn"] = "Sigrún"

vinir["SR"]["Sími"] = "799-2345"

Þar sem við höfum tvenns konar vísa í sama fylkinu, **vinir**, er sagt að þetta sé fylki með tvær víddir.

Þegar við höfum sett upp forritið getur forritið tekið við skammstöfun vinar sem inntaki og síðan skrifað út upplýsingarnar sem við höfum geymt um þá. Hér er heilt forrit sem gerir þetta:

vinir["AJ"]["Nafn"] = "Anna"

vinir["AJ"]["Sími"] = "799-6789"

vinir["BÞ"]["Nafn"] = "Björn"

vinir["BÞ"]["Sími"] = "799-4567"

vinir["SR"]["Nafn"] = "Sigrún"

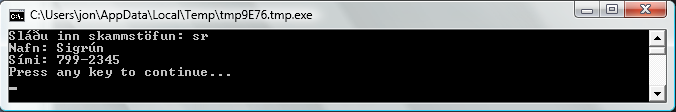
vinir["SR"]["Sími"] = "799-2345"

TextWindow.Write("Sláðu inn skammstöfun: ")

skammst = TextWindow.Read()

TextWindow.WriteLine("Nafn: " + vinir[skammst]["Nafn"])

TextWindow.WriteLine("Sími: " + vinir[skammst]["Sími"])



Mynd 52 – Einföld símaskrá

# Notkun fylkja fyrir töflur

Mjög algeng not fyrir fylki með fleiri en eina vídd er að geyma upplýsingar fyrir hnitakerfi, töflur og þess háttar. Slík fyrirbæri hafa línur og dálka sem geta fallið vel inn í tvívítt fylki. Hér fyrir neðan er einfalt forrit sem raðar mislitum boxum í grind:

línufj = 8

dálkafj = 8

stærð = 40

For l = 1 to línufj

For d = 1 To dálkafj

GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()

**box[l][d]** = Shapes.AddRectangle(stærð, stærð)

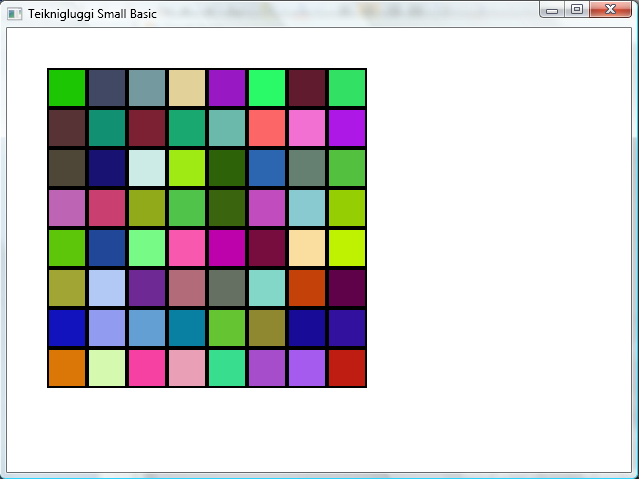
Shapes.Move(box[l][d], d \* stærð, l \* stærð)

Program.Delay(300)

EndFor

EndFor

Þetta forrit býr til box og raðar þeim í 8 x 8 stafla (gæti líka verið tafla eða hnitakerfi). Auk þess sem það teiknar boxin geymir það þau í fylki. Það auðveldar okkur að henda reiður á þeim og að nota þau aftur eftir þörfum. Tímatöfin er sett inn svo hægt sé að sjá boxin staflast.



Mynd 53 – Boxum staflað

Þar sem við höfum geymt boxin í fylki gætum notað þau aftur með því að lesa þau úr fylkinu. Þannig getum við bætt eftirfarandi kóða aftan við forritið hér á undan til að láta boxin færast úr staflanum upp í vinstra hornið.

For l = 1 To línufj

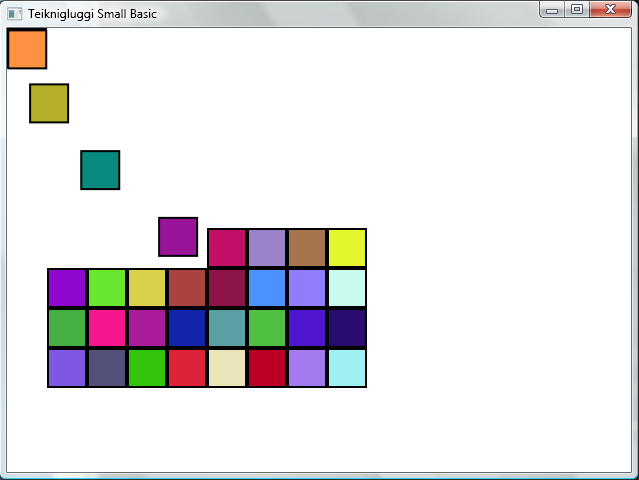
For d = 1 To dálkafj

Shapes.Animate(**box[l][d]**, 0, 0, 1000)

Program.Delay(300)

EndFor

EndFor



Mynd 54 – Unnið með boxin í grindinni

Kafli 11

Atvik og gagnvirkni

Í fyrstu tveimur köflunum kynntum við hluti sem hafa *eiginleika* (enska: properties) og *aðgerðir* (enska: operations). Auk eiginleika og aðgerða hafa sumir hlutir það sem kallað er ***atvik*** (enska: events). Atvik eru eins og nemar sem gefa merki, til dæmis sem viðbrögð við einhverju sem notandinn gerði, svo sem að hreyfa músina eða ýta á músarhnapp. Að vissan hátt eru atvik andstæðan við aðgerðir. Forritarinn notar aðgerðir til að láta tölvuna gera eitthvað, en það er tölvan sem notar atvik til að láta forritið þitt vita að eitthvað sem skiptir máli hafi gerst.

# Hvernig geta atvik verið gagnleg?

Atvik eru í aðalhlutverki þegar kemur að því að gera forrit gagnvirk. Ef þú vilt að notandinn eigi samskipti við forritið þitt eru atvik það sem þú notar. Segjum að þú sért að forrita leik. Þú vilt auðvitað að forritið skynji hvað spilarinn gerir. Til þess notar þú atvik – forritið tekur við gjörðum notandans með því að nota atvik. Ekki hafa áhyggjur þó þetta virðist vera flókið, því við ætlum að skoða einfalt forrit sem hjálpar þér að skilja hvað atvik eru og hvernig er hægt að nota þau.

Hér fyrir neðan er mjög einfalt forrit með aðeins einni skipun og einni stefju. Stefjan notar *ShowMessage* aðgerð *GraphicsWindow* hlutarins til að birta skilaboðaglugga fyrir notandann.

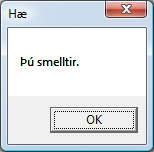
GraphicsWindow.MouseDown = onMouseDown

Sub onMouseDown

GraphicsWindow.ShowMessage("Þú smelltir.", "Hæ")

EndSub

Athyglisverðasti hluti forritsins hér fyrir ofan er línan þar sem tengjum heiti stefju við **MouseDown** atvik *GraphicsWindow* hlutarins. *MouseDown* líkist mjög eiginleika, en í stað þess að eitthvert gildi sé sett í það er það tengt við stefjuna *onMouseDown*, sem framkvæmir eitthvað. Það er þetta sem er sérstakt við atvik – þegar atvikið á sér stað, er kallað sjálfkrafa á stefjuna. Í þessu tilfelli er kallað á stefjuna *onMouseDown* í hvert skipti sem notandinn ýtir á músarhnapp innan teiknigluggans. Keyrðu forritið til að sjá hvernig þetta virkar. Í hvert skipti sem þú smellir með músinni, birtist skilaboðagluggi eins og sá sem myndin sýnir.



Mynd 55 – Viðbrögð við atviki

Að nota atvik á þennan hátt er mjög öflugt og gerir mögulegt að búa til skapandi og áhugaverð forrit. Forrit sem eru skrifuð á þennan hátt eru oft kölluð atvikaknúin forrit.

Það er hægt að breyta *onMouseDown* stefjunni svo hún geri annað en birta skilaboð. Til dæmis gætum við teiknað stórar blár doppur þar sem notandinn smellir, eins og gert er í þessu forriti.

GraphicsWindow.BrushColor = "Blue"

GraphicsWindow.MouseDown = OnMouseDown

breidd = 20

hæð = 20

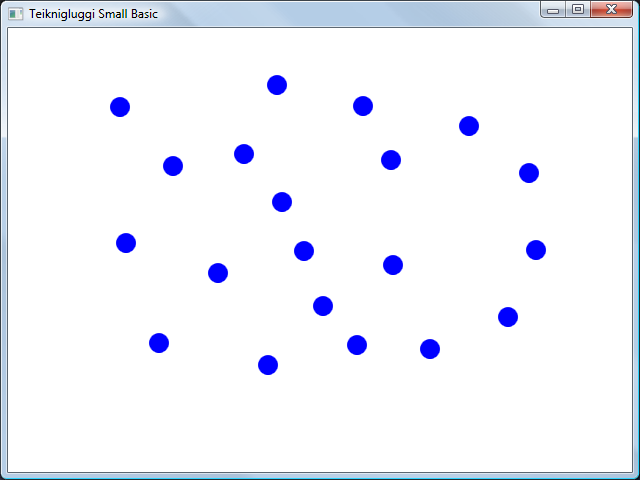
Sub OnMouseDown

x = GraphicsWindow.MouseX - breidd / 2

y = GraphicsWindow.MouseY - hæð / 2

GraphicsWindow.FillEllipse(x, y, breidd, hæð)

EndSub



Mynd 56 – Brugðist við músarhnappsatviki

Taktu eftir því að í forritinu hér fyrir ofan notum við *MouseX* og *MouseY* eiginleikana til að sækja hnit músarinnar. Við drögum síðan hálfa breidd og hæð hringsins frá hnitunum áður en hringurinn er teiknaður svo að miðja hringsins birtist þar sem músin er.

# Unnið úr mörgum atvikum

Það eru í raun engin takmörk fyrir því hversu mörg atvik þú getur notað í forriti. Þú getur jafnvel látið sömu stefjuna bregðast við mörgum atvikum. En þú getur aðeins brugðist einu sinni við hverju atviki. Ef þú reynir að tengja tvær stefjur við sama atvik, hefur sú seinni vinninginn.

Til sýna fram á að það sé hægt að nota fleiri en eitt atvik skulum við taka dæmið hér á undan og bæta við stefju sem keyrir þegar ýtt er á einhvern lykil. Við skulum láta þessa nýju stefju skipta um lit í penslinum, þannig að þegar þú smellir músinni þá hefur doppan fengið nýjan lit.

GraphicsWindow.BrushColor = "Blue"

GraphicsWindow.MouseDown = OnMouseDown

GraphicsWindow.KeyDown = OnKeyDown

breidd = 20

hæð = 20

Sub OnKeyDown

GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()

EndSub

Sub OnMouseDown

x = GraphicsWindow.MouseX – breidd / 2

y = GraphicsWindow.MouseY – hæð / 2

GraphicsWindow.FillEllipse(x, y, breidd, hæð)

EndSub



Mynd 57 – Brugðist við fleiri en einu atviki

Ef þú keyrir þetta forrit og smellir á gluggann færðu bláa doppu. En ef þú ýtir á einhvern lykil og smellir svo aftur, færðu doppu í öðrum lit. Það sem gerist þegar þú ýtir á lykil er að atvikið *KeyDown* á sér stað og keyrir stefjuna *onKeyDown* en hún setur einhvern lit af handahófi í pensilinn. Eftir það þegar þú smellir músinni er doppa teiknuð í nýja litnum og sá litur helst þangað til þú ýtir aftur á lykil og nýr litur er valinn af handahófi. *KeyDown* atvikið á sér stað alveg óháð *MouseDown* atvikinu.

# Teikniforrit

Með atvik og stefjur á valdi okkar, getum við nú skrifað forrit sem gerir mögulegt fyrir notendur að teikna í glugganum. Það er auðveldara en maður gæti ímyndað sér að skrifa slíkt forrit ef við gætum þess að skipta verkefninu upp í smærri einingar. Sem fyrsta þrep skulum við skrifa forrit sem lætur músina skilja eftir sig slóð þegar notandinn hreyfir músina í glugganum.

GraphicsWindow.MouseMove = OnMouseMove

Sub OnMouseMove

x = GraphicsWindow.MouseX

y = GraphicsWindow.MouseY

GraphicsWindow.DrawLine(prevX, prevY, x, y)

prevX = x

prevY = y

EndSub

En þegar forritið er keyrt byrjar fyrsta línan alltaf í horni gluggans efst til vinstri (hniti 0,0) vegna þess að *prevX* og *prevY* hafa gildin 0 í upphafi. Við getum lagað þetta forrit með því að bregðast við *MouseDown* atvikinu og setja gildi í *prevX* og *prevY* þegar það atvik á sér stað (þegar ýtt er á músarhnapp).

Auk þess þurfum við aðeins að fá slóð þegar notandinn heldur músarhnappnum niðri. Annars ætti línan ekki að teiknast. Til að fá þessa hegðun notum við *IsLeftButtonDown* eiginleika **Mouse** hlutarins. Þessi eiginleiki segir til um hvort vinstri hnappi músarinnar er haldið niðri eða ekki. Ef þessi eiginleiki hefur gildið True, á að teikna línu, annars á að sleppa henni.

GraphicsWindow.MouseMove = OnMouseMove

GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown

prevX = GraphicsWindow.MouseX

prevY = GraphicsWindow.MouseY

EndSub

Sub OnMouseMove

x = GraphicsWindow.MouseX

y = GraphicsWindow.MouseY

If (Mouse.IsLeftButtonDown) Then

GraphicsWindow.DrawLine(prevX, prevY, x, y)

EndIf

prevX = x

prevY = y

EndSub

Viðauki A

Nokkur dæmi til gamans

# Trítla teiknar brotamynd



Mynd 58 – Trítla teiknar tré

horn = 30

munur = 10

vegalengd = 60

Turtle.Speed = 9

GraphicsWindow.BackgroundColor = "Black"

GraphicsWindow.PenColor = "LightGreen"

TeiknaTré()

Sub TeiknaTré

If (vegalengd > 0) Then

Turtle.Move(vegalengd)

Turtle.Turn(horn)

Stack.PushValue("vegalengd", vegalengd)

vegalengd = vegalengd - munur

TeiknaTré()

Turtle.Turn(-horn \* 2)

TeiknaTré()

Turtle.Turn(horn)

vegalengd = Stack.PopValue("vegalengd")

Turtle.Move(-vegalengd)

EndIf

EndSub

# Myndir af Flickr



Mynd 59 – Myndir sóttar af Flickr

GraphicsWindow.BackgroundColor = "Black"

GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown

mynd = Flickr.GetRandomPicture("mountains, river")

GraphicsWindow.DrawResizedImage(mynd, 0, 0, 640, 480)

EndSub

# Lifandi bakgrunnur skjáborðs

For i = 1 To 10

mynd = Flickr.GetRandomPicture("mountains")

Desktop.SetWallPaper(mynd)

Program.Delay(10000)

EndFor

# Spaðaleikur



Mynd 60 – Leikur með spaða

GraphicsWindow.BackgroundColor = "DarkBlue"

spaði = Shapes.AddRectangle(120, 12)

bolti = Shapes.AddEllipse(16, 16)

GraphicsWindow.MouseMove = OnMouseMove

x = 0

y = 0

deltaX = 1

deltaY = 1

RunLoop:

x = x + deltaX

y = y + deltaY

gw = GraphicsWindow.Width

gh = GraphicsWindow.Height

If (x >= gw - 16 or x <= 0) Then

deltaX = -deltaX

EndIf

If (y <= 0) Then

deltaY = -deltaY

EndIf

padX = Shapes.GetLeft (spaði)

If (y = gh - 28 and x >= padX and x <= padX + 120) Then

deltaY = -deltaY

EndIf

Shapes.Move(bolti, x, y)

Program.Delay(5)

If (y < gh) Then

Goto RunLoop

EndIf

GraphicsWindow.ShowMessage("Þú tapaðir", "Spaðaleikur")

Sub OnMouseMove

spaðiX = GraphicsWindow.MouseX

Shapes.Move(spaði, spaðiX - 60, GraphicsWindow.Height - 12)

EndSub

Viðauki B

Litir

Hér er listi yfir liti sem Small Basic styður flokkaður eftir grunnlitnum.

## Rauðir litir

|  |  |
| --- | --- |
| IndianRed | #CD5C5C |
| LightCoral | #F08080 |
| Salmon | #FA8072 |
| DarkSalmon | #E9967A |
| LightSalmon | #FFA07A |
| Crimson | #DC143C |
| Red | #FF0000 |
| FireBrick | #B22222 |
| DarkRed | #8B0000 |

## Bleikir litir

|  |  |
| --- | --- |
| Pink | #FFC0CB |
| LightPink | #FFB6C1 |
| HotPink | #FF69B4 |
| DeepPink | #FF1493 |
| MediumVioletRed | #C71585 |
| PaleVioletRed | #DB7093 |

## Appelsínugulir litir

|  |  |
| --- | --- |
| LightSalmon | #FFA07A |
| Coral | #FF7F50 |
| Tomato | #FF6347 |
| OrangeRed | #FF4500 |
| DarkOrange | #FF8C00 |
| Orange | #FFA500 |

## Gulir litir

|  |  |
| --- | --- |
| Gold | #FFD700 |
| Yellow | #FFFF00 |
| LightYellow | #FFFFE0 |
| LemonChiffon | #FFFACD |
| LightGoldenrodYellow | #FAFAD2 |
| PapayaWhip | #FFEFD5 |
| Moccasin | #FFE4B5 |
| PeachPuff | #FFDAB9 |
| PaleGoldenrod | #EEE8AA |
| Khaki | #F0E68C |
| DarkKhaki | #BDB76B |

## Fjólubláir litir

|  |  |
| --- | --- |
| Lavender | #E6E6FA |
| Thistle | #D8BFD8 |
| Plum | #DDA0DD |
| Violet | #EE82EE |
| Orchid | #DA70D6 |
| Fuchsia | #FF00FF |
| Magenta | #FF00FF |
| MediumOrchid | #BA55D3 |
| MediumPurple | #9370DB |
| BlueViolet | #8A2BE2 |
| DarkViolet | #9400D3 |
| DarkOrchid | #9932CC |
| DarkMagenta | #8B008B |
| Purple | #800080 |
| Indigo | #4B0082 |
| SlateBlue | #6A5ACD |
| DarkSlateBlue | #483D8B |
| MediumSlateBlue | #7B68EE |

## Grænir litir

|  |  |
| --- | --- |
| GreenYellow | #ADFF2F |
| Chartreuse | #7FFF00 |
| LawnGreen | #7CFC00 |
| Lime | #00FF00 |
| LimeGreen | #32CD32 |
| PaleGreen | #98FB98 |
| LightGreen | #90EE90 |
| MediumSpringGreen | #00FA9A |
| SpringGreen | #00FF7F |
| MediumSeaGreen | #3CB371 |
| SeaGreen | #2E8B57 |
| ForestGreen | #228B22 |
| Green | #008000 |
| DarkGreen | #006400 |
| YellowGreen | #9ACD32 |
| OliveDrab | #6B8E23 |
| Olive | #808000 |
| DarkOliveGreen | #556B2F |
| MediumAquamarine | #66CDAA |
| DarkSeaGreen | #8FBC8F |
| LightSeaGreen | #20B2AA |
| DarkCyan | #008B8B |
| Teal | #008080 |

## Bláir litir

|  |  |
| --- | --- |
| Aqua | #00FFFF |
| Cyan | #00FFFF |
| LightCyan | #E0FFFF |
| PaleTurquoise | #AFEEEE |
| Aquamarine | #7FFFD4 |
| Turquoise | #40E0D0 |
| MediumTurquoise | #48D1CC |
| DarkTurquoise | #00CED1 |
| CadetBlue | #5F9EA0 |
| SteelBlue | #4682B4 |
| LightSteelBlue | #B0C4DE |
| PowderBlue | #B0E0E6 |
| LightBlue | #ADD8E6 |
| SkyBlue | #87CEEB |
| LightSkyBlue | #87CEFA |
| DeepSkyBlue | #00BFFF |
| DodgerBlue | #1E90FF |
| CornflowerBlue | #6495ED |
| MediumSlateBlue | #7B68EE |
| RoyalBlue | #4169E1 |
| Blue | #0000FF |
| MediumBlue | #0000CD |
| DarkBlue | #00008B |
| Navy | #000080 |
| MidnightBlue | #191970 |

## Brúnir litir

|  |  |
| --- | --- |
| Cornsilk | #FFF8DC |
| BlanchedAlmond | #FFEBCD |
| Bisque | #FFE4C4 |
| NavajoWhite | #FFDEAD |
| Wheat | #F5DEB3 |
| BurlyWood | #DEB887 |
| Tan | #D2B48C |
| RosyBrown | #BC8F8F |
| SandyBrown | #F4A460 |
| Goldenrod | #DAA520 |
| DarkGoldenrod | #B8860B |
| Peru | #CD853F |
| Chocolate | #D2691E |
| SaddleBrown | #8B4513 |
| Sienna | #A0522D |
| Brown | #A52A2A |
| Maroon | #800000 |

## Hvítir litir

|  |  |
| --- | --- |
| White | #FFFFFF |
| Snow | #FFFAFA |
| Honeydew | #F0FFF0 |
| MintCream | #F5FFFA |
| Azure | #F0FFFF |
| AliceBlue | #F0F8FF |
| GhostWhite | #F8F8FF |
| WhiteSmoke | #F5F5F5 |
| Seashell | #FFF5EE |
| Beige | #F5F5DC |
| OldLace | #FDF5E6 |
| FloralWhite | #FFFAF0 |
| Ivory | #FFFFF0 |
| AntiqueWhite | #FAEBD7 |
| Linen | #FAF0E6 |
| LavenderBlush | #FFF0F5 |
| MistyRose | #FFE4E1 |

## Gráir litir

|  |  |
| --- | --- |
| Gainsboro | #DCDCDC |
| LightGray | #D3D3D3 |
| Silver | #C0C0C0 |
| DarkGray | #A9A9A9 |
| Gray | #808080 |
| DimGray | #696969 |
| LightSlateGray | #778899 |
| SlateGray | #708090 |
| DarkSlateGray | #2F4F4F |
| Black | #000000 |