Internetguide #40
Kom igång med Scratch!

Programmera ett spel, steg för steg i Scratch.

Måns Jonasson

I den här guiden lär du dig...

- ☑ Grunderna i Scratch
- ☑ Använda block
- ☑ Skapa ett enkelt program
- ☑ Använda villkor
- ☑ Lägga in ljud
- ☑ Lägga in bakgrunder
- ☑ Skapa ett spel där du kombinerar olika funktioner

Innehåll

Del 2 – Bläckfiskspelet 48

Välkommen till Kom igång med Scratch!

Målet med den här kursen är att du ska få en första inblick i hur det är att programmera egna spel och andra program. Idag är nästan alla barn i Sverige vana vid att använda datorer, surfplattor och mobiltelefoner. De kan installera appar, spela spel, skriva skolarbeten, använda webbläsare och söka på nätet. Det är dock inte så många barn som fått en inblick i hur det är att själv instruera sina smarta prylar och få dem att göra vad som helst.

När hemdatorerna blev vanliga på 1980-talet uppstod en generation av ”sovrumsprogrammerare” – vanliga barn och ungdomar som lärde sig att programmera sina datorer, dels eftersom det inte fanns samma utbud av färdiga program och spel som idag, men antagligen främst för att alla hemdatorer kom förprogrammerade med BASIC – ett enkelt programmeringsspråk som lockade många till att börja experimentera med programmering på ett enkelt sätt.

Dagens PC-datorer har oftast inte någon programmeringsmiljö installerad från början och många av enheterna som våra barn använder är av säkerhetsskäl slutna plattformar, vilket gör det närmast omöjligt för en vanlig användare att börja skapa egna program.

Om den här kursen

Den här instruktionsboken är skriven för den som är absolut nybörjare när det kommer till programmering. Den hör ihop med Internetguiden ”Barnhack - kom igång med programmering!”.

Vad är Scratch?

Scratch är en mjukvara framtagen av MIT (Massachusetts Institute of Technology) för att introducera programmering som koncept för barn och ungdomar. Eftersom det är lätt att komma igång med Scratch är det dock också många vuxna som gillar att göra egna program med hjälp av Scratch.

Den som börjar leka med Scratch lär sig snabbt grunderna i programmering som koncept. Du använder dig av objekt, variabler, loopar, villkor och mycket annat som blir en solid grund att bygga vidare på om du blir sugen på att fortsätta din programmeringskarriär efter att ha börjat med Scratch. Det finns stora mängder färdiga spel och program till Scratch att titta på, lära sig av och börja ändra i.

Barn som använder Scratch blir snabbt kompetenta ”mini-programmerare” och kan redan efter en stund göra egna spel och program. Men kanske allra viktigast är att barnen lär sig att datorerna tar sina instruktioner från människor, och att alla som vill kan lära sig hur.

Grunderna i Scratch – vad är vad?

Scratch började som ett Java-program, men inför version 2.0 byggde man om hela programmet till Adobe Flash. Den senaste versionen används alltså i din webbläsaren genom att surfa till: http://scratch.mit.edu

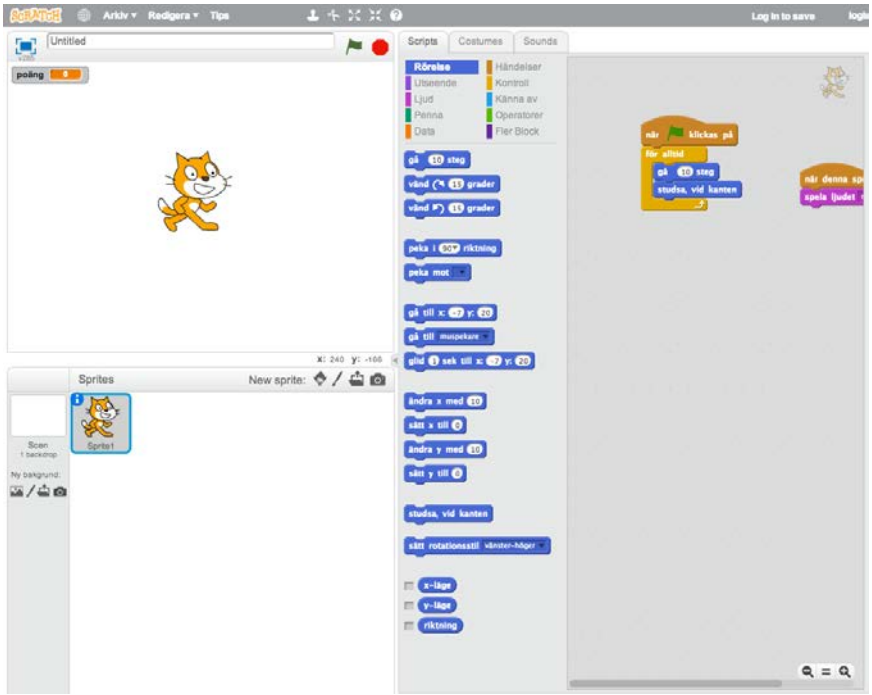
Det du möts av där är ett antal färdiga Scratch-program som du kan titta på för att se vad som går att göra med Scratch, men det du är intresserad av är så klart att börja skapa ditt eget program. Klicka därför på ”Skapa” högst upp i menyn (Om webbplatsen är på engelska när du kommer in står det ”Create” där i stället).

Det som händer nu är att editorn i Scratch laddas över hela webbläsarens yta. Du går igenom de olika delarna av editorn först av allt.

Scratch finns översatt till en mängd olika språk. Börja med att växla språk i editorn till svenska genom att klicka på ”globen” i menyraden och välja svenska där.

Miljön i Scratch är indelad i några olika områden. Överst till vänster har du en yta som kallas ”scen”. Under den visas de ”sprites” som du har att arbeta med. Längst till höger visas de ”script” som är kopplade till en vald sprite, och i mitten har du ett bibliotek med de olika script som finns att använda, indelade i avdelningar med olika typer av ”block”.

Editorn i Scratch 2.0



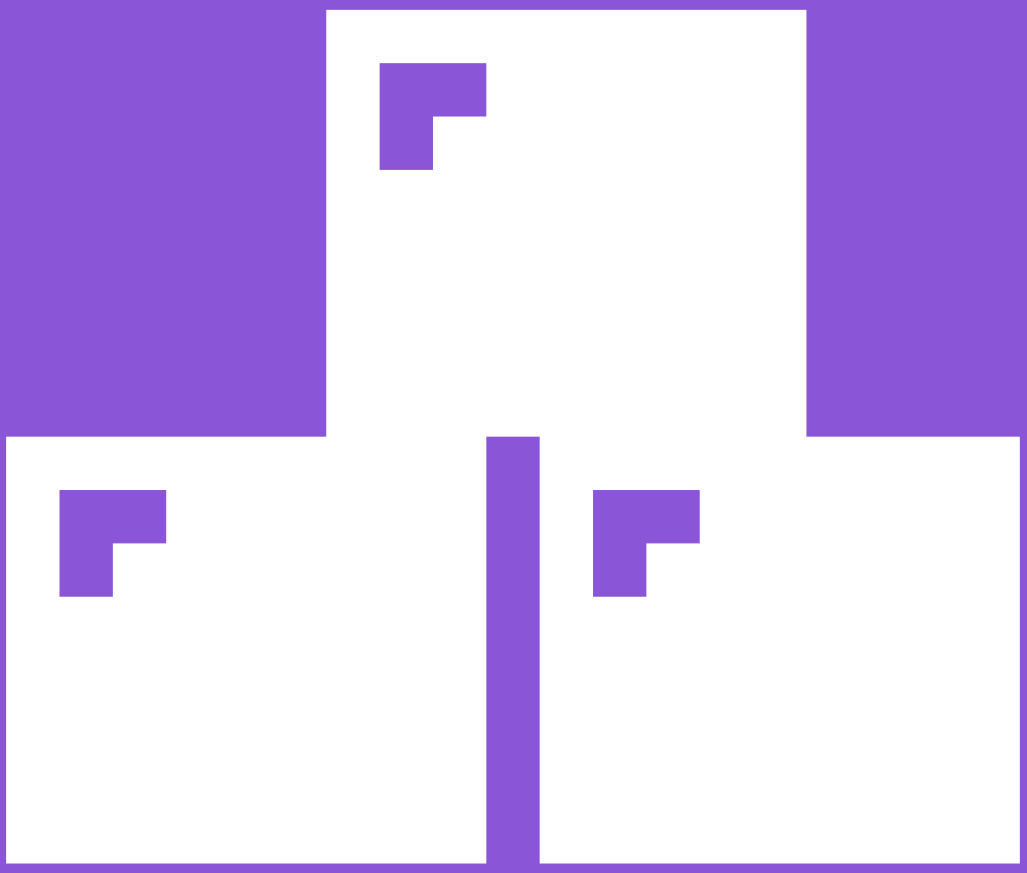
Man kan tänka på Scratch som en teater. Teatern har en scen där allting utspelar sig. Scenen kan byta dekor (bakgrund) och på scenen ställer sig en eller flera skådespelare (sprites) för att agera. Skådespelarna agerar enligt sitt manus (script).

I Scratch är det du som är regissör, rollbesättare, manusförfattare, scenograf och kostymör! Du bestämmer vilken pjäs som ska spelas, vilka som ska vara med i den, vad skådespelarna ska säga, hur scenen ska se ut och vad skådespelarna ska ha på sig.

För att datorn ska kunna utföra dina instruktioner och programmet ska köras som du tänkt dig behöver du koppla ihop dina sprites med olika block till ett script och placera dem på scenen. Du kan se vilka script som är kopplade till en sprite genom att antingen klicka på spriten i din sprite-lista eller genom att dubbelklicka på en sprite som är placerad på scenen. När du först börjar använda en sprite har den inga script kopplade till sig, så det ser tomt ut i ytan till höger. För att instruera en sprite drar du block från sprite-biblioteket till scriptytan till höger, vilket kopplar ett block till just den spriten. Mer om det senare, för nu är det dags att sluta snacka och börja göra lite program!

Nu kastar du dig rakt in i Scratch, och du kanske börjar undra över några av de saker som gås igenom i det här kapitlet. Bli inte orolig för det, utan försök ändå att bara följa kapitlet från början till slut. Du kommer att lära dig några av de grundläggande sakerna kring hur Scratch fungerar, och du kommer att gå in på detaljerna senare.

Block – script i olika kategorier



Överst i mitten, ovanför blockbiblioteket, ser du tre flikar – ”Script”, ”Klädslar” och ”Ljud”. Under dessa ligger tio kategorirubriker med olika färgkoder.

De tio olika scriptkategorierna i Scratch

Du kommer nu att titta lite närmare på de olika kategorierna av script-block, men du börjar med att titta på hur själva blocken ser ut:

Blocket som rör en sprite ett antal steg

Förutom färgen på blocket ovan så ser du att det har en liten inbuktning på ovansidan och en utbuktning på undersidan. Det här är Scratch sätt att visa dig hur de olika blocken kan sättas ihop, precis som pusselbitar. Just det här blocket kan du alltså sätta ihop med ett block från ovansidan och ett annat block från undersidan. På så sätt kan du styra i vilken ordning instruktionerna ska utföras, eftersom Scratch läser ett script uppifrån och ner. Det block du sätter under detta kommer att läsas efter att Scratch gjort klart det som ditt block ska göra.

Rörelse

Nu kan du titta på kategorierna. Först ligger ”rörelse”. Här hittar du de block som används för att flytta en sprite på scenen. Det allra första blocket, som du tittade på nyss, flyttar en sprite 10 steg framåt. Du ser att siffran ”10” ligger i en vit oval. Det betyder att du kan klicka där och ändra siffran från 10 till vad du vill – och Scratch kommer att flytta denna sprite så många steg som du angett när blocket körs. Här finns en mängd olika block och du kommer att kunna använda fler av dem lite längre fram.

Utseende

Under ”utseende” ligger lila block som styr hur en sprite eller scenen ska se ut. Här finns till exempel möjligheten att gömma sprites ett tag, eller att ändra bakgrund på scenen.

Ljud

Under ”ljud” ligger förstås de block som låter Scratch spela upp ljud av olika slag. Om en sprite ska skrika ”Aj” när den träffas av något är det med hjälp av de rosa ljudblocken du gör det.

Penna

De gröna ”penna”-blocken låter oss rita på scenen. Du kan göra ett eget ritprogram i Scratch om du vill och de här blocken hjälper till.

Data

Under ”data” är det nästan helt tomt när du börjar med ett nytt projekt i Scratch. Men här finns ett par knappar, nämligen ”Skapa en variabel” och ”Skapa en lista” som du kommer att kunna använda för att skapa små virtuella lådor att spara saker i. Mer om det senare.

Händelser

Den här kategorin är en av de viktigaste, för det är genom att koppla händelseblock till dina sprites som du kan få dem att reagera när du till exempel klickar på dem med muspekaren eller trycker ner en tangent på tangentbordet.

Kontroll

De gula kontrollblocken är också väldigt viktiga. Här kan du skapa loopar, villkor och annat som låter dig styra hur och när dina sprites ska agera.

Känna av

De här ljusblå blocken har flera olika former, vilket du lär dig mer om lite senare. Här finns block som låter en sprite reagera om den till exempel rör vid en annan sprite, vilket är väldigt användbart till exempel när du gör spel och vill att något ska hända om två sprites krockar med varandra.

Operatorer

Här ligger allt som har med matte att göra, tillsammans med en hel del andra gröna block som hanterar beräkningar och logik. Ibland behöver du till exempel kunna jämföra två tal med varandra, och då är det med operatorer du kan göra det.

Fler block

Den här kategorin är överkurs för tillfället. I Scratch 2.0 kan man göra helt egna block, men funktionen är inte riktigt färdig än, så du hoppar över den till en annan gång.

Ditt första program – Du säger hej till katten Scratch

Katten Scratch

Katten är bara en av väldigt många olika figurer som följer med programmet, men eftersom han står där mitt på scenen redan när editorn öppnas kan du ju börja med att se till att han har något att göra.

I den här övningen kommer du använda ett par olika block för att få katten att springa över skärmen och dessutom jama när du klickar på honom med muspekaren. Kanske inte världens roligaste program, men du kommer att lära dig ett par grundläggande saker innan du börjar lära dig mer om de olika kategorierna av script-block.

Infoknappen finns på varje sprite

Varje sprite kan finnas på två ställen när den laddats in i Scratch. Katten, som är laddad från början, står dels mitt på scenen, dels återfinns han längre ner på skärmen, i sprite-listan där alla dina laddade sprites alltid ligger. I övre vänstra hörnet av varje sprite i listan finns en blå rund info-knapp. Här klickar du för att ställa olika egenskaper för en sprite, så klicka på den nu för att se alternativen för katten, eller ”Sprite1” som den heter just nu.

Inställningar för sprite

I rutan som visas nu kan du se några olika detaljer för just den sprite du klickade på, en instans av katten Scratch. Först och främst går det att döpa en sprite till något som går lättare att känna igen, om du vill. Under namnet finns några värden: x, y och riktning. X och y är spritens position på scenen, och från allra första början när katten står mitt på scenen är dess position X:0 och Y:0. Du kommer att lära dig mer om positioner i koordinater senare, men nästa värde, ”riktning”, är viktigt redan nu.

Riktningen säger åt vilket håll en sprite pekar. I stället för att säga att en sprite pekar åt exempelvis höger eller neråt används grader. Noll grader är rakt uppåt, så 90° betyder att den här spritens riktning just nu är rakt åt höger på scenen, vilket också visas av den lilla ratten till höger om siffran. Prova att klicka på ratten och dra med musen åt olika håll medan du håller ner musknappen. Du ser dels att siffran ändras mellan noll och 360 grader, men du ser också att den katt som visas på scenen snurrar runt för att följa med din inställning på ratten. Ställ tillbaka riktningen till 90 grader just nu, för du vill att katten ska börja gå åt höger när han sätter igång.

På nästa rad finns något som heter ”rotationsstil”. Det här berättar för editorn hur spriten ska tillåtas att rotera. Som standard kan en sprite rotera i alla riktningar, vilket du såg alldeles nyss när du drog i riktningsratten. Här finns tre inställningar: ”full rotation”, ”vänster-höger” och ”ingen rotation” som bara ser ut som en prick. Klicka på pricken för att bestämma att spriten inte får rotera alls. Testa nu igen att dra i riktningsratten ovanför och du ser att även om riktningssiffran ändras så står katten still och snurrar inte alls. Det är för att du ställt in denna sprite på att inte roteras alls. Ställ in ”rotationsstil” på ”vänster-höger” nu genom att klicka på mittenalternativet som ser ut som ett rakt streck med två pilspetsar och se till att riktningen ovanför är på 90 grader igen om du ställde om den för att testa nyss. Nu har du ställt in din sprite på att bara få vända sig rakt åt höger eller vänster, inga andra håll.

Den sista rutan i inställningspanelen för vår sprite heter ”kan dras på scenen” och den bestämmer huruvida en sprite kan flyttas med hjälp av musen när programmet körs senare. Du låter den vara urkryssad för katten, du vill ju flytta honom med hjälp av script, inte med musen.

För att komma tillbaka till sprite-listan, klicka på den lilla runda blå knappen med en vänsterpil precis till vänster om katten (se bilden ovan).

Ditt första script – låt katten springa

För att katten ska kunna agera måste du ge den ett manus – ett script. När du klickar på en sprite i sprite-listan så visas just den spritens script i den stora grå ytan till höger på skärmen som du kallar ”sprite-script”. Där är tomt just nu eftersom katten inte fått några block kopplade till sig än, men du ser en liten utsuddad miniversion av spriten längst upp till höger för att påminna oss om vilken sprite du jobbar med för tillfället.

Nu ska du få katten att springa över skärmen! För att göra detta behöver du koppla ett händelseblock och ett rörelseblock till katten.

Om du tittar till vänster i editorn, i det övre högra hörnet av scenen, så ser du en grön flagga och en röd cirkel där. Det här är Scratchs start- och stoppknappar.

Scratchs start- och stoppknappar

Klicka nu på ”Händelser” överst i blockbiblioteket för att lista alla bruna block som har med händelser att göra. För att få saker att börja hända kan du använda det allra första händelseblocket: ”När [grön flagga] klickas på”.

När [grön flagga] klickas på

Det här blocket är ett av startblocken i Scratch, för som du ser finns det ingen inbuktning ovanpå, så det går inte att haka det här blocket på undersidan av andra block. Däremot finns det en utbuktning på undersidan, vilket är en inbjudan till att lägga till andra block efter.

När du kopplar det här händelseblocket till en sprite så gör du det möjligt att låta spriten agera så fort du klickar på knappen med den gröna flaggan ovanför scenen. För att koppla blocket till din katt-sprite drar du helt enkelt blocket från biblioteket in i sprite-script-ytan till höger. Du kan släppa det var som helst, men just nu kan du lägga det någonstans i övre delen av scriptet.

Nu har du berättat för Scratch att du vill att katten, ”Sprite1”, ska agera när du klickar på startknappen. Men om du klickar på startknappen nu händer ingenting, för du har ju inte berättat vad som ska hända när du klickar på den. För att se till att något händer måste du välja ett annat block och haka på det på undersidan av ditt händelseblock.

Klicka på kategorin ”Rörelse” i blockbiblioteket för att lista alla blå rörelseblock. På det allra första blocket står det ”Gå (10) steg”.

Rörelseblock: gå (10) steg

Varje gång det här blocket körs kommer den sprite som det är kopplat till att röra sig ett antal steg i den riktning som spriten pekar. Din katt pekar ju just nu mot 90 grader, rakt åt höger, så om det här blocket körs kommer katten att gå tio steg åt höger. Det står ”10” i rutan från början, men det går att ändra till vad du vill. Det struntar du i just nu, för tio steg är lagom.

Klicka och dra det blå blocket till scriptytan, och håll koll på vad som händer när du börjar närma dig ditt bruna händelseblock som redan ligger där.

Scratch visar att blocken kommer att passa ihop

Ser du att när du för det blå blocket mot underkanten av ditt bruna händelseblock så lyser kanten upp? Det här är Scratchs sätt att visa för dig att de här blocken kan sitta ihop. Om du släpper det blå blocket medan den vita kanten lyser kommer Scratch att sätta ihop blocken till ett script. Släpp det blå blocket så att det passar ihop med det bruna händelseblocket. Om du missar gör det inget, du kan ta tag i det blå blocket igen och passa in det bättre, tills det fastnar där det ska.

Två block bildar ett script

Grattis! Du har skapat ditt allra första script. Testa nu att klicka på startknappen. Katten hoppar tio steg åt höger, men stannar där. Scratch gör alltså precis som du sagt åt det – när knappen med den gröna flaggan klickas på så går katten tio steg framåt. Men du ville ju att katten skulle springa fram över skärmen, så nu måste du berätta för Scratch att du vill att katten ska fortsätta att springa, inte bara hoppa fram tio steg en enda gång när du klickar på startknappen.

Loopar – så gör du saker flera gånger

Att se till så att ett eller flera block körs flera gånger i rad automatiskt kräver att du skapar en ”loop”. Det finns olika typer av loopar, eller framförallt finns det olika sätt att kontrollera hur looparna startar och slutar, men för tillfället ska du skapa en oändlig loop – något som ska utföras under hela tiden vårt program körs, från det att du trycker på startknappen tills du trycker på stoppknappen.

Två block bildar ett script

Klicka på den gula kategorin ”Kontroll” i blockbiblioteket. Du ser en mängd gula block som alla handlar om olika sätt att kontrollera flödet i ett script. Nu ska du använda ett block som skapar en loop, och det ser lite annorlunda ut. Det har fortfarande en inbuktning på ovansidan, så det går att haka på efter andra block. Men det har ingen utbuktning på undersidan, så det går inte att haka på fler block efter. Varför då? Jo, när Scratch kommer till det här blocket så kommer det att börja köra alla block som ligger inuti detta block, om och om igen i evighet. Eller åtminstone tills du trycker på startknappen.

Blocket ”För alltid”

Som du ser har det här blocket alltså en öppning på mitten, vilket gör att du kan lägga ett eller flera block inuti det. De block som ligger inuti körs i loopen, så länge loopen körs.

Det du vill göra är ju att se till så att katten springer över skärmen hela tiden så fort du trycker på startknappen. Du ska alltså lägga det blå blocket ”gå 10 steg” inne i en loop, så att Scratch förstår att du vill att blocket ”gå 10 steg” ska köras om och om igen. För att göra detta måste du först plocka isär det blå blocket ”gå 10 steg” från det bruna blocket ”När [grön flagga] klickas på”. Att ta isär block är lika lätt som att sätta ihop dem. Ta tag i det blå blocket och dra det lite åt sidan. Du kan lägga det bredvid det bruna blocket i scriptytan, bara de inte sitter ihop längre.

Nu går det bra att dra in vårt gula loop-block ”för alltid” och sätta ihop det med det bruna ”När [grön flagga] klickas på”. Gör som tidigare, dra in det nya blocket tills du ser den vita lysande kanten och släpp så blocken sätts ihop.

Startblocket och loopblocket sitter ihop, men det blå blocket är lagt åt sidan så länge

Nu kan du ta tag i det blå blocket ”gå 10 steg” igen och dra in det i loopen. Se till att du släpper det när det blir en vit kant vid utbuktningen inuti loopen ”för alltid”, se bilden:

Släpp blocket inuti loopen

Nu har du skapat din första loop och i den har du lagt blocket ”gå 10 steg”. Det du gjort är berätta för Scratch att du vill att blocket ”gå 10 steg” ska köras om och om igen, ”för alltid”, när startknappen klickats.

Din första loop

Testa att klicka på den gröna flaggan nu och se hur katten springer iväg hela vägen till den högra kanten av scenen och stannar allra längst ut, när han nästan försvunnit ur bild. Scratch har gjort som du sagt, men det blev inte riktigt bra ändå, för när katten kom till kanten av scenen tog det stopp. Varför då?

Scratch låter inte dina sprites försvinna bort från scenen, för då skulle du inte kunna ta tag i dem med musen och dra dem dit du vill senare. Därför struntar Scratch i din instruktion ”gå 10 steg” när katten inte kan komma längre framåt. Det du behöver göra nu är att få katten att vända när han kommer till kanten. Som tur är har Scratch ett färdigt blått block för att göra just det, och det ligger också under kategorin ”Rörelse”. Blocket du ska använda nu heter ”studsa, vid kanten”.

Blocket ”studsa, vid kanten”

Vad det här blocket gör är att helt enkelt låta en sprite studsa när det når fram till kanten av scenen. Att en sprite studsar betyder egentligen att den vänder 180 grader om. Eftersom din katt hade

riktningen 90 grader, rakt åt höger, till att börja med kommer den nu att få riktningen -90 grader, eller rakt åt vänster när den studsar. Det här kanske låter rörigt, men egentligen behöver du inte bry dig om antalet grader på riktningen just nu, så länge du minns att ”studsa, vid kanten” låter din sprite vända helt om i motsatt riktning när den träffar kanten.

För att Scratch ska kunna utföra studsen behöver du ta tag i det blå blocket ”studsa, vid kanten” och lägga det inuti loopen. Varför inne i loopen? Jo, för om du lägger blocket utanför loopen kommer Scratch bara att köra blocket en enda gång, direkt efter det att du klickat på startknappen. Om katten inte råkar befinna sig vid scenkanten just då kommer den inte att studsa, och när du sen flyttar på katten med ”gå 10 steg” inuti loopen finns där inget block som studsar katten när det behövs.

Ta alltså tag i det blå blocket ”studsa, vid kanten” och lägg det inuti loopen. Det spelar ingen roll om du lägger blocket före eller efter det blå blocket ”gå 10 steg”, men lägg det efter just nu. Ditt färdiga script ska nu se ut så här:

Ditt första script är färdigt!

Klicka på startknappen en gång till. Nu börjar katten gå fram och tillbaka, fram och tillbaka på scenen. Varje gång den är på väg att springa bort vid scenkanten så kommer den att studsa och börja gå åt andra hållet.

Om du tittar på ditt script medan katten springer så ser du att hela scriptet liksom glöder, och omgärdas av en ljusgul färg. Det är Scratch som visar dig att det scriptet körs för tillfället. När du börjar göra större program kommer du kanske att ha flera script som ligger bredvid varandra och körs vid olika tillfällen. Markeringen runt det script som körs är ett bra sätt att hålla koll på vad det är som händer för tillfället i Scratch.

Scriptet glöder när det körs

Okej, katten springer fram och tillbaka på scenen, men finns det något du kan göra för att det hela ska kännas lite mer som ett spel? Du kanske ska ta och göra så att katten jamar till när man klickar på den?

Flera script till samma sprite

Det finns inget som hindrar att du skapar flera script till samma sprite. Nu har du ett script som gör så att katten springer. Men om du vill att katten ska jama när du klickar på den kan du skapa ett till script till katten och lägga det bredvid ditt första script. Scriptytan är ganska stor, så det finns plats för många script om du vill skapa flera.

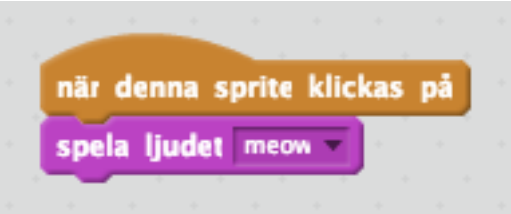
Klicka på kategorin ”Händelser” igen. Här ligger ditt gamla bekanta bruna block ”När [grön flagga] klickas på”. Men här ligger också ett annat händelsblock, nämligen ”När denna sprite klickas på”. Det här blocket vill du ha nu, för detta block körs bara när du klickar på spriten med musen. Dra in det bruna blocket ”När denna sprite klickas på” och lägg den någonstans i scriptytan, en bit ifrån ditt första script.

Du sätter igång med ett till script bredvid ditt första.

Ljud – låt katten jama

Det här kommer att bli ett mycket enklare script, för det enda du vill göra är att spela upp ett ”mjau” när du klickar på spriten. Klicka på den rosa kategorin ”Ljud” och leta efter blocket ”spela ljudet [meow]”. Dra in det rosa blocket ”spela ljudet [meow]” och haka fast det under det bruna blocket ”när denna sprite klickas på”. Som du ser ligger ordet ”meow” i en liten ruta med en pil. Det finns flera block som har samma utseende, och gemensamt för dem är att den lilla rutan låter dig välja mellan flera alternativ, om det finns. Scratch börjar med att bara ha ett ljud laddat, nämligen ”meow”, så just nu går det inte att välja något annat ljud, men det passar ju bra eftersom du vill att katten ska jama.

Ditt andra script är färdigt



Ditt andra script är redan färdigt. Klicka på startknappen och försök att träffa katten med ett klick med muspekaren medan den springer på scenen. När du träffar katten så jamar den!

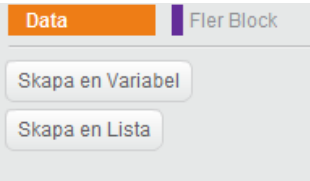
Variabler – håll något i minnet

Det här börjar ju likna ett spel! Men vore det inte lite roligare om du fick en poäng varje gång du fick katten att jama? För att räkna poäng måste du låta Scratch hålla en siffra i minnet, nämligen själva poängen. Du vill att poängen ska börja på noll och sen räknas upp med ett varje gång du träffar katten.

För att ett program ska kunna hålla saker i minnet finns något som kallas ”variabler”. Variabler kan man tänka på som ett slags lådor som du kan lägga olika saker i. När du lagt något i lådan kan du syssla med annat ett tag, men när du tittar i lådan igen ligger dina saker kvar. Du kan lägga till eller ta bort saker ur lådan, och Scratch håller ordning på innehållet åt dig. En sådan låda blir utmärkt att använda för att hålla koll på poängräkningen i ditt allra första spel.

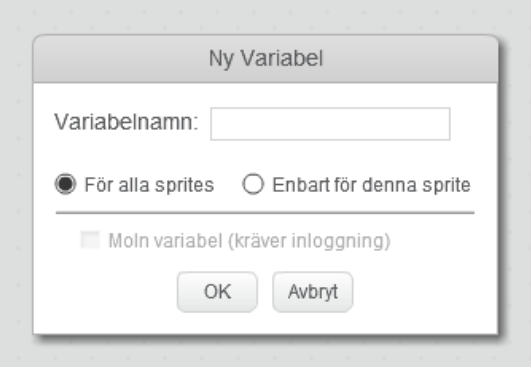
Scratch kan hantera massor av variabler samtidigt. Just nu behöver du bara använda en variabel, och för att använda den måste du först skapa den. Klicka på kategorin ”Data” så ser du att här finns inga block just nu, men däremot ett par knappar.

Knapparna för att skapa variabler



Du vill skapa en ny variabel, så klicka på ”Make a variable”. Genast blir det lite svårare, för nu tycker Scratch att du ska svara på ett par frågor:

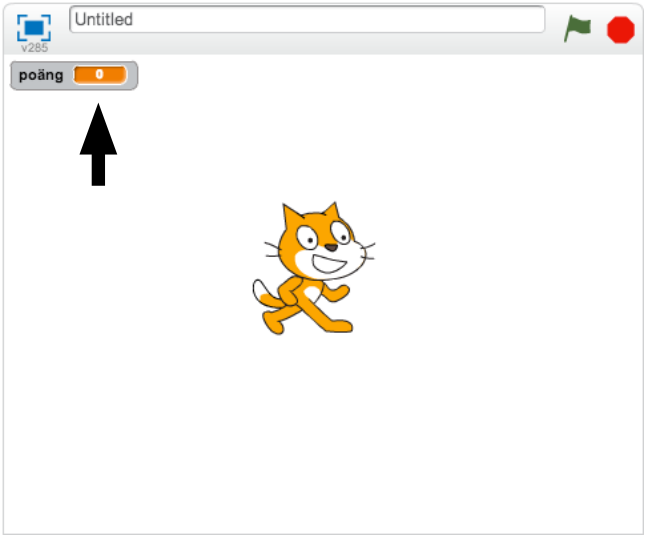
Variabelfrågor



Först av allt vill Scratch att du döper vår nya variabel. Det är enda sättet som Scratch kan hålla koll på olika lådor senare, så skriv in ”poäng” i fältet för variabelnamn. Nästa val handlar om vilka sprites som ska få använda den här variabeln. För tillfället spelar det ingen roll, så låt det valet vara och klicka på ”OK” för att skapa variabeln ”poäng”.

Genast händer två saker. Det första du ser är att det nu plötsligt dök upp ett antal orange block som alla har med din nya variabel att göra. Dessutom dök en ny liten ruta upp längst upp till vänster på scenen:

Din nya variabel visas på scenen



Det här är ju perfekt för dig, för du vill ju visa poängen hela tiden medan spelet pågår. Men om du tittar i blockbiblioteket så ser du att vår nya variabel ligger överst i en orange oval, och att det finns en liten kryssruta till vänster om den:

Variabeln du skapat

Testa att kryssa ur rutan till vänster om din variabel så ser du att poängrutan nu försvinner från scenen. Kryssa i rutan igen så kommer den tillbaka på samma ställe. På detta sätt kan du själv välja om dina variabler ska visas på scenen eller inte. Man kan använda variabler på många olika sätt. En del variabler, som din nya poängvariabel, är viktiga för den som använder programmet att ha koll på. Den som spelar ditt nya spel vill ju veta hur många poäng den har. Men många variabler behövs bara inuti själva programmet för att hålla koll på olika saker. Då är det onödigt att visa dem för användaren, och du kommer att göra mer av det längre fram.

Att räkna upp eller ner variabler

Nu har du alltså en låda att spara spelarens poäng i, men om du sätter igång programmet nu så ser du att poängen aldrig räknas upp oavsett hur många gånger du klickar på katten. För att Scratch ska veta när poängen ska räknas upp måste du berätta när och hur det ska göras.

Du har ju redan ett script som håller koll på när katten träffas av muspekaren. Det scriptet gör så att Scratch spelar upp ljudet ”meow” när man klickar på katten. I det scriptet blir det perfekt att också lägga in ett block som räknar upp poängen.

Under ”Data” ligger nu ett antal nya block som alla har att göra med din variabel. Bland dem ser du det orange blocket ”ändra [poäng] med 1”. Ta det blocket och dra in det till vårt script, under det rosa blocket ”spela ljudet [meow]”.

Poänguppräkning på plats

Det du nu gjort är att säga åt Scratch att ”när denna sprite klickas på”, ”spela ljudet [meow]” och ”ändra [poäng] med 1”. Det där ”ändra [poäng] med 1” betyder att du ska ta det tal som för tillfället ligger i variabeln ”poäng” och lägga till ett. Som du ser ligger ettan i en vit boll, och det betyder att du kan skriva in vilken siffra som helst där. Om du vill att man ska få tio eller tusen poäng varje gång man klickar på katten går det att lägga in siffrorna ”10” eller ”1000” där i stället. För tillfället låter du det ge en poäng i taget att klicka på katten, så låt ettan stå kvar.

Nu kan du testa att köra igång programmet igen, och klicka på katten. Förutom att katten jamar när du träffar honom med muspekaren (stackars katt!) så får du ett poäng för varje träff, vilket också visas direkt i den lilla rutan för din variabel ”poäng” på scenen.

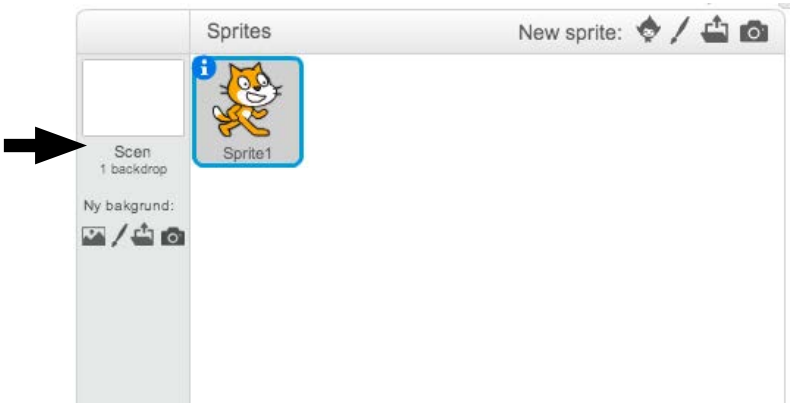
Här finns en viktig sak att förstå. Att det står ”poäng” bredvid din poängsiffra på scenen är bara för att du döpte variabeln till ”poäng” när du skapade den. Du hade lika gärna kunnat döpa den till ”sillar” eller ”jamningar” eller vad du vill. Då hade det stått i stället bredvid siffran på scenen.

Scenen kan också ha script

Du får ett poäng varje gång du träffar katten, men om du missar katten händer ju inget, så du kan klicka hej vilt för att få poäng utan att det gör något. Du kanske skulle ta och straffa spelaren om han eller hon missar katten och bara klickar bredvid?

I ditt första program är scenen rätt tråkig – bara en vit ruta. Men scenen kan ha egna scripts och det kommer du att testa nu. För att visa scenens script klickar du på rutan för ”Scen” till vänster om sprite-listan:

Klicka här för att visa scenens script



Nu blir scriptytan tom, för scenen har inga script än. För att visa kattens script igen klickar du bara på ”Sprite1” till höger, och på så sätt kan du växla mellan scenens script och de olika spritens script medan du skapar nya program.

I scenens script-yta ska du nu lägga ett script som tar bort en poäng varje gång man klickar på scenen. För om du klickar på scenen har du ju missat katten. Börja med att välja den bruna kategorin ”Händelser” och dra in startblocket ”när denna sprite klickas på”. Namnet på detta block är något förvirrande nu, för det är ju ingen sprite du klickar på utan scenen, men blocket heter likadant ändå.

Nu har du början på ett script som kommer att köras varje gång man klickar på scenen. Under kategorin ”Data” har du din gamla bekant, det orange blocket ”ändra [poäng] med 1”. Dra in det blocket och haka fast det under ”när denna sprite klickas på”. Som det ser ut nu skulle du alltså få ytterligare en poäng i din variabel om du klickar på scenen, men du vill ju dra av en poäng i stället. Det är lätt gjort – klicka på siffran ”1” i den vita bollen i blocket och skriv ”-1”. Samma

block används alltså för att både lägga till och dra bort poäng, och det är vad du skriver in i den vita bollen som avgör hur värdet på variabeln ändras.

Dra av en poäng när scenen klickas på

Kör igång programmet igen och testa att klicka på katten och att klicka på scenen. Som du ser så minskar poängen när du klickar på scenen, men du ökar poängen när du träffar katten.

Initiering – hur ska du ha det när du börjar?

Spelet är nästan klart, men om du klickat på katten och scenen några gånger nu så är vår poängsiffra antingen plus eller minus någon siffra. Du vill att poängen ska vara noll när spelet börjar, så att spelaren inte får börja spelet med en massa minuspoäng i bagaget, till exempel.
Att ställa in variabler och andra saker i ett program när det körs kallas att initiera programmet. Du vill kanske nollställa poängräknare, placera saker på rätt ställe eller göra andra inställningar så att alla spelare får samma förutsättningar varje gång ett nytt spel påbörjas. Du har ju redan ett script som körs när grön flagga klickas på, och det ligger kopplat till ”Sprite1”, det vill säga katten. Klicka på katten i sprite-listan för att se kattens script igen.

I ditt lilla spel räcker det ju om du ser till att variabeln ”poäng” sätts till noll när spelet börjar. Under kategorin ”Data” finns det orange blocket ”Sätt [poäng] till 0”. Dra in det till scriptet som har startblocket ”när [grön flagga] klickas på” och se till att släppa det innan loopen men efter startblocket:

Placera det nya blocket mellan startblocket och loopblocket

Varför lägger du inte det här blocket inne i loopen? Jo, du vill ju bara att poängen ska nollställas när spelet först startar, alltså när du klickar på den gröna flaggan. Om du lägger nollställningsblocket inne i loopen kommer poängen att sättas till noll oavbrutet, och det spelar ingen roll hur mycket du försöker räkna upp eller ner poängen – den sätts genast till noll ändå.

Det här är ett sätt att göra det här på. Du hade också kunnat skapa ett tredje script med samma startblock som ditt första – ”när [grön flagga] klickas på” och lägga nollställningsblocket ensamt där. Kom ihåg att en sprite kan ha många script, och att det kan finnas flera script som har samma händelseblock som start.

Ett alternativ till initiering

När du senare gör program som har många och långa script kan det vara en god idé att hålla isär script som gör olika saker, men just nu spelar det inte så stor roll, så du kan välja vilken av de två lösningarna du vill.

Kör igång spelet nu genom att trycka på den gröna flaggan. Poängen sätts genast till noll och du kan börja spela genom att jaga katten. När du avslutar spelet och startar om det kommer poängställningen alltid att börja från noll.

Ljud – allt blir roligare med musik

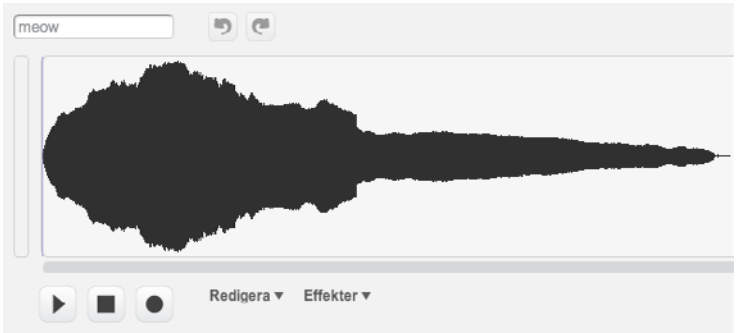
Du har redan använt ett ljudblock för att få katten att jama när du klickar på den, men det finns mycket mer du kan göra med ljud-blocken i Scratch. Klicka på kategorin ”Ljud” så ser du att här finns många block som har med instrument och taktslag att göra. Men du såg ju förut att det bara var ”meow” som gick att välja i ljudblocket, så hur får du fler ljud i Scratch?

Script, Costumes, Sounds

Över blockkategorierna finns tre flikar. Först har du ”Script”, och det är den som är vald när du startar Scratch. Nästa kategori heter ”Costumes” och den går vi igenom om lite senare, för just nu är det ”Sounds” som är intressant. Klicka på den.

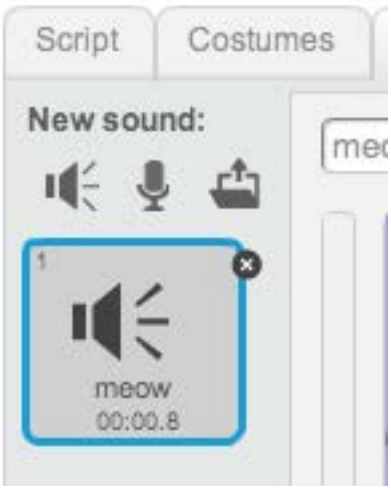
Här ser du nu ljudet ”meow” som kommer förladdat i Scratch. Du ser hela ljudet som en ljudvåg och du kan lyssna på det genom att trycka på play-knappen.

Ljudet ”meow”



När du vill ha fler ljud i Scratch använder du den lilla menyn som nu har ersatt blockkategorierna och där finns det tre knappar att välja mellan:

Ljudmenyn

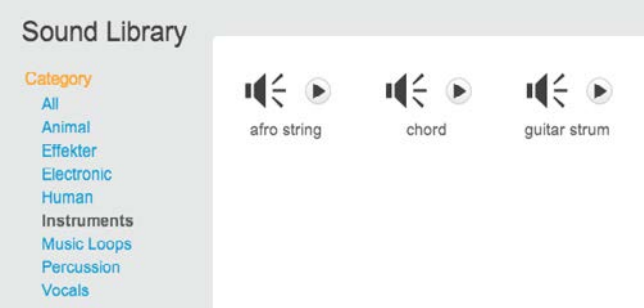


Den första knappen låter dig ladda in ett ljud från Scratchs inbyggda ljudbibliotek. Den andra knappen låter dig spela in ett eget ljud med datorns mikrofon. Och den tredje knappen låter dig ladda upp en ljudfil från din egen dator. För tillfället ska du försöka att hitta ett ljud från Scratchs medföljande ljudbibliotek, så tryck på den första knappen, den som ser ut som en högtalare.

Ladda in ljud från det medföljande biblioteket

Du får upp en lista med massor av ljud, och till vänster har du möjlighet att filtrera ljud efter olika kategorier. Klicka på ”Instruments” i listan till vänster.

Ljudbiblioteket, filtrerat på kategorin ”Instruments”



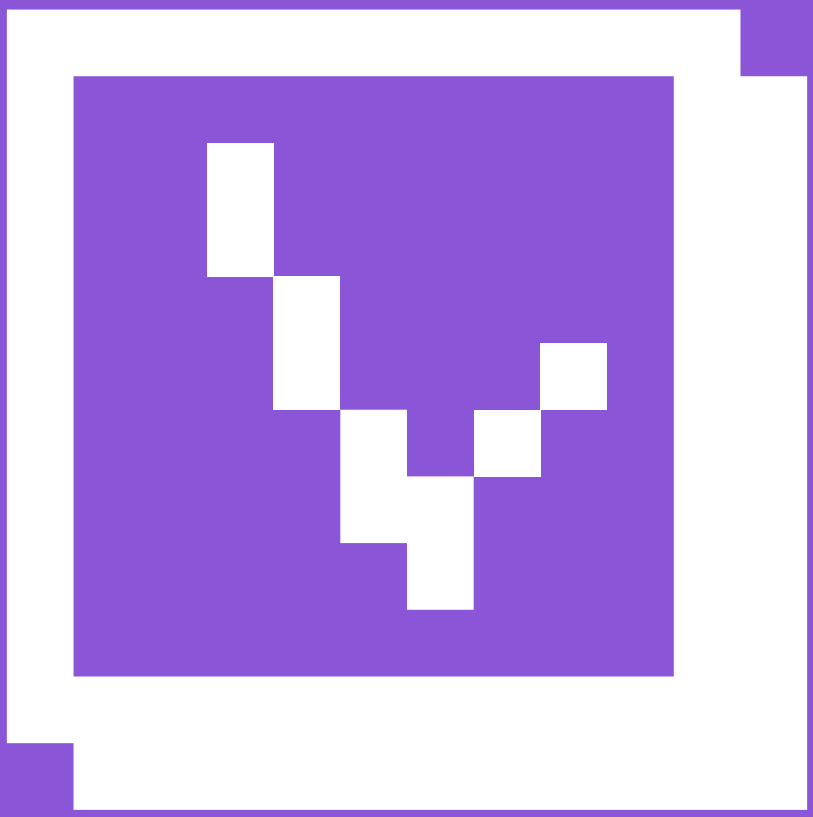
Genom att trycka på den lilla play-knappen bredvid varje ljud kan du provlyssna på alla ljuden. Du kan använda det ljud som heter ”guitar strum” som grattisljud när spelet är slut, så dubbelklicka på det ljudet, eller klicka på det en gång för att markera och sen på knappen ”OK” längst ner till höger.

Ljudet ”Guitar strum” laddat

När ljudet laddats hamnar det i ljudlistan under ”meow” som ju redan låg där innan. På det här sättet kan du ha massor av ljud inne i Scratch och använda dem vid olika tillfällen i dina program.

Klicka på fliken ”Script” för att visa blockbiblioteket och scriptytan igen.

Villkor – kör bara koden om något stämmer



Väldigt ofta vill du att ett block eller ett helt script bara ska köras om något annat händer, eller bara om en variabel har ett visst värde. Du kanske har gjort ett spel och vill att något ska hända varje gång du trycker på mellanslagstangenten, eller så vill du ge spelaren minuspoäng om en sprite träffar en annan. Det finns flera sätt att hålla koll på att saker körs bara under vissa förutsättningar, och det handlar om villkor.

Under kategorin ”Kontroll” ligger alla block som hjälper dig att bara köra script när vissa förutbestämda saker inträffat. Det viktigaste och mest använda av dem heter ”om < > då”. Det du kan göra med det blocket är att först kolla efter något – till exempel värdet på en variabel – och endast om det stämmer med vad du vill, gå vidare och köra de block som ligger inuti ”om < > då”. Som du ser har ”om < > då” en diamantformad ruta i mitten. Det finns nämligen ett antal block i Scratch som är just diamantformade, och det är bara de som passar att lägga till i ”om < > då”.

Blocket ”om < > då”

Du kan fundera på några exempel på när det vore intressant att hålla koll på ett villkor. Exempelvis om du vill att ditt spel med katten ska ta slut när man fått 10 poäng. Då skulle du kunna använda blocket ”om < > då” för att säga något i stil med ”OM (värdet på variabeln poäng = 10) DÅ ... gör nånting!” Men hur kollar du då om värdet på din variabel är tio? Det här kräver ett antal steg. Titta först under den gröna kategorin ”Operatorer”. Här finns massor av block, men det du behöver för att jämföra två värden är just diamantformat och heter ”[] = []”:

Operatorn ”[] = []”

Som du ser kan du skriva in egna värden här, men det är ju ganska meningslöst att skriva in två fasta värden. Riktigt användbart är detta block för att du kan släppa ett variabelblock i ena hålet. Titta under kategorin ”Data” så ser du att vår variabel ”poäng” har ett helt eget, ovalt orange block där. Det blocket passar i endera hålet i det gröna operatorblocket, och det blocket i sin tur passar i det gula ”om < > då”:

Variabelblock inuti operatorblock inuti kontrollblock...

Nu kan du skriva in siffran ”10” i det högra hålet, och plötsligt har du en kombination av block som tillsammans bildar ett villkor. Om poäng = 10, då ska någonting hända.

Om poäng = 10 då

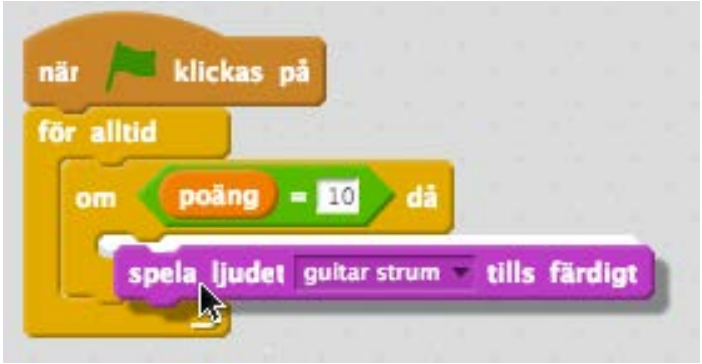
För att använda den här blockkombinationen måste du lägga den i ett script som körs. I ditt spel kan du lägga den i ett nytt script, och för att det ska kollas hela tiden behöver du lägga det i en loop som körs för alltid:

Script för att kolla om poäng = 10

Det här scriptet kommer att börja köras när den gröna flaggan klickas på, och loopen som körs ”för alltid” ser till att Scratch hela tiden kollar om poäng är 10, och då... Ja, vadå? Än så länge är det tomt inuti vårt villkorsblock. Du kan väl spela en liten fanfar för spelaren? En sån tur att du redan laddat in ljudet ”guitar strum”!

Det finns ett annat block för att spela upp ljud som är användbart här. Det heter ”spela ljudet [] tills färdigt”, men eftersom Scratch gissar att du troligen vill använda det senaste ljudet som laddats in är ditt nya ljud redan förvalt, så det står ”spela ljudet [guitar strum] tills färdigt” på det just nu. Det fina med det blocket är att det spelar upp ljudet och väntar snällt tills ljudet har spelat klart. Dra in det blocket och lägg det inuti ditt villkor:

Släpp ljudblocket inuti villkorsblocket



De block som ligger inuti vårt villkorsblock är ju de som kommer att köras bara när villkoret är uppfyllt. Om du skulle lägga det här ljud-blocket utanför villkorsblocket skulle det köras ”för alltid” och det skulle bli en hel gitarrkonsert under spelets gång i stället för bara när spelaren vunnit.

Nu kan du testa att köra spelet igen. När du fått tio poäng börjar gitarren att spela. Men som du hör slutar den aldrig, utan spelar om och om igen. Det är nämligen precis vad du sagt åt Scratch att göra. Om poäng = 10 då ska ljudet spelas. Eftersom villkoret ligger inne i en loop som körs ”för alltid” kommer villkoret att vara sant så länge poängen är tio.

Det kanske vore bra om ljudet spelades, och sen fick spelaren börja om spelet från början igen? Du kan lägga till en liten paus också, så att spelaren hinner förstå vad som händer. Och katten kan väl säga ”Grattis, du vann”? Du fortsätter att bygga inuti villkorsblocket.

Under kategorin ”Kontroll” finns blocket ”vänta (1) sekunder”. Dra in det blocket och lägg det efter ”spela ljudet [guitar strum] tills färdigt”. Ändra så att Scratch väntar tre sekunder i stället.

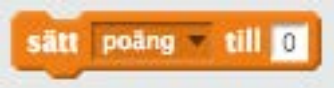
Vänta tre sekunder efter att ljudet spelats färdigt

Nu spelar Scratch alltså ett ljud och väntar tre sekunder när spelaren når tio poäng. Nu vill du att katten ska berätta för spelaren att hon eller han vunnit spelet, och för det ändamålet har Scratch fina pratbubblor inbyggda. Under kategorin ”Utseende” finns det lila blocket ”säg [] i () sekunder”.

Ändra texten från ”Hello!” till ”Grattis, du vann!” genom att klicka i den vita rutan och markera texten. Och ändra antalet sekunder till fyra så spelaren hinner läsa.

Scratch har spelat ljudet, väntat några sekunder och sen låtit katten säga att spelaren vunnit. Men för att ditt villkor ska sluta vara sant hela tiden och därmed bara fortsätta att köra blocken inuti om och om igen måste du se till att variabeln poäng inte längre är tio. Under kategorin ”Data” finns det orange blocket ”Sätt [] till []”:

Blocket ”sätt [] till []”



Just nu är det redan ifyllt på just det vis du vill, nämligen att variabeln ”poäng” ska sättas till noll, så dra in det här blocket och lägg det sist inuti ditt villkorsblock. Det här blocket kan du annars använda för att sätta vilken variabel som helst till vilket värde som helst, så det är ett mycket användbart block.

Nu har du gjort klart scriptet som sköter spelets slut:

Det färdiga villkorsscriptet

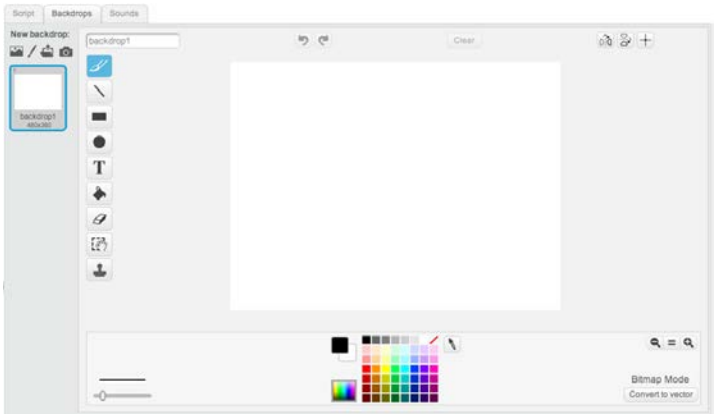
Testa ditt spel nu, och se vad som händer när poängen når tio. Du får en liten trudelutt, katten säger att du vunnit och sen börjar det om igen eftersom poängen sätts till noll.

Bakgrunder (klädslar) – scenens dekor

Katten i ditt spel springer fram och tillbaka på scenen men det ser ju ut som att han gör det i snön, för scenen är bara vit. Tack och lov har Scratch möjlighet att lägga in fina bakgrundsbilder på scenen, och spelet kommer att bli mycket snyggare när du lägger in en miljö på scenen.

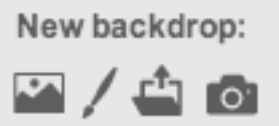
Klicka på fliken ”Backdrops” ovanför blockbiblioteket för att visa dina bakgrunder.

Backdrops



Det här är en rolig del av Scratch, för här inne gömmer sig ett helt litet ritprogram. Även om katten ser ut att springa mitt i ingenting springer han faktiskt på en bakgrundsbild, även om den är helt vit från början. Om du vill kan du använda penseln eller de andra målar-verktygen och färgerna här för att rita din egen bakgrund.

Menyn för ny bakgrund

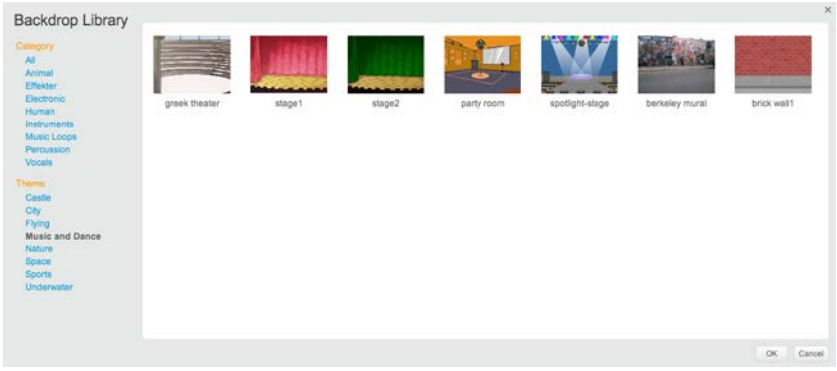


Överst till vänster i den här vyn hittar du valen för ny bakgrund. De är ganska lika valen som finns för ljud. Först är det något som ser ut som en tavla. Här får du välja bland de medföljande bilderna att ladda in. Sen kommer en liten pensel, och den knappen skapar en

ny, tom och vit bakgrund att måla på. Den tredje knappen låter dig ladda upp en bild från din egen dator att använda som bakgrund. Den sista knappen är en liten kamera, och när du klickat på den kan du använda din dators webbkamera (om du har någon) för att ta ett foto att använda som bakgrund.

Just nu ska du testa de medföljande bakgrunderna, så klicka på den första knappen, den som ser ut som en liten tavla. Nu öppnar sig biblioteket med massor av bakgrunder att välja på, och till vänster kan du precis som i ljudbiblioteket filtrera på olika kategorier. Välj kategorin ”Musik och Dans”.

Bakgrunder filtrerat på ”Music and Dance”



Ladda in bakgrunden som heter ”brick wall1” genom att antingen dubbelklicka på den eller först klicka på den en gång för att markera den och sen klicka på ”OK” längst ner till höger.

Genast laddas bakgrunden in och läggs den på scenen. Nu står din sprite med katten ovanpå scenen med en bakgrundsbild, och kanske ligger katten inte riktigt rätt – för om den inte står på rätt ställe ser det ut som att katten flyger ovanför marken:

Bakgrund laddad, men katten flyger!

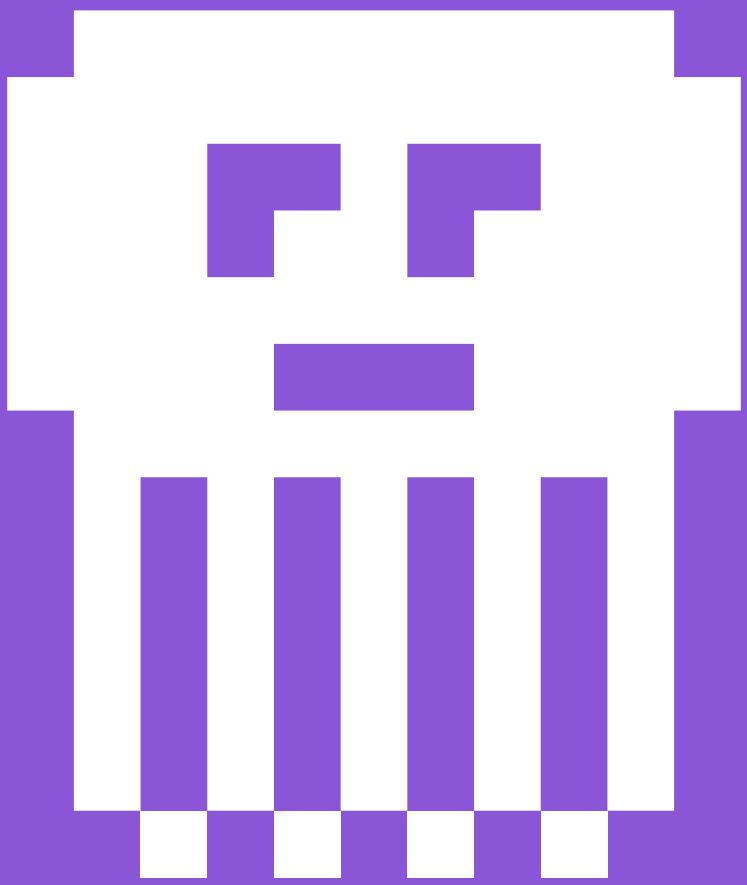
Det är lätt åtgärdat. Ta bara tag i katten med muspekaren och dra den till rätt höjd så att det ser ut som om katten står på gatan i stället. Om spelet är igång blir det svårt, så stoppa spelet först med stoppknappen.

Nu springer katten fram och tillbaka på gatan och dina script funkar precis som förut.

Fortsättning följer

Du har nu gjort ett första, väldigt enkelt spel med Scratch. Under tiden har du lärt dig om sprites och script, loopar och villkor, operatorer och variabler. Det finns väldigt mycket du kan göra med bara olika varianter av de block du använt hittills. Du kan rita egna bakgrunder och sprites. Du kan spela in egna ljud och göra spel eller interaktiva filmer. Men Scratch kan mycket, mycket mer än så.

Del 2 – Bläckfiskspelet



Under mina Barnhack använder jag innehållet i den första kursen för att lära ut grunderna i Scratch under en timme. Den andra timmen låter jag sedan barnen fortsätta på egen hand. De brukar då antingen fortsätta med kattspelet, eller så börjar de på ett eget spel medan jag går runt och hjälper till. Vissa frågor brukar komma upp varje gång, och dem har jag med i den här fortsättningen. Dessutom brukar jag visa lite olika typer av andra spel som är enkla att bygga i Scratch för att sporra kreativiteten i rummet, och även de spelen finns med i den här delen.

Jag hoppas att ni blir inspirerade att fortsätta med Scratch och att ni delar ut era bästa spel i Scratch-communityn!

Bläckfiskspelet

I denna övning ska vi göra ett litet undervattensspel – vi ska styra en bläckfisk som simmar fram och tillbaka över havsbotten, försöka fånga stjärnor för att få poäng samtidigt som vi måste akta oss för hajar och krabbor som lurar överallt!

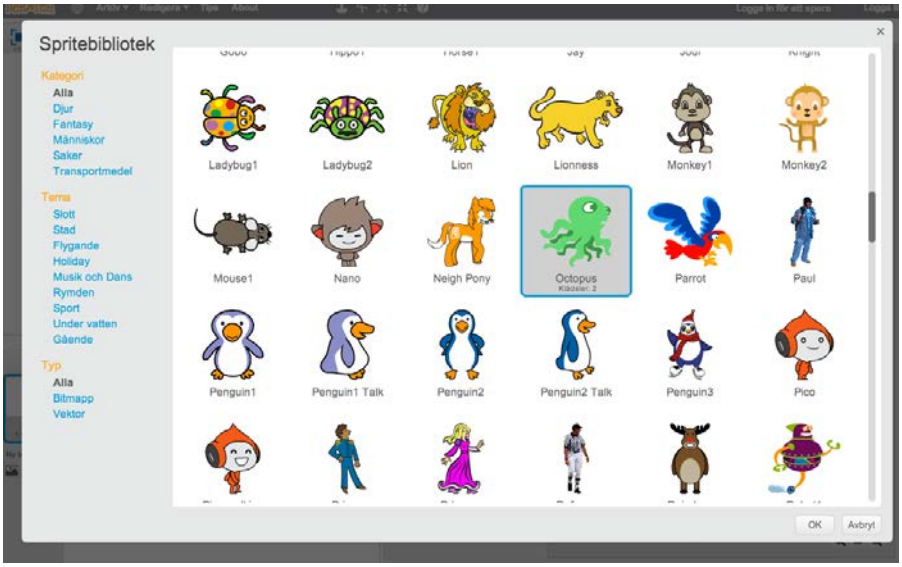
Att använda piltangenterna för att styra en sprite

I det första kattspelet vi byggde lät vi katten springa fram och tillbaka av sig själv medan vi jagade poäng genom att klicka på katten. Nu är det dags att se hur vi kan använda piltangenterna för att styra en sprite, och dessutom testar vi lite andra figurer än katten!

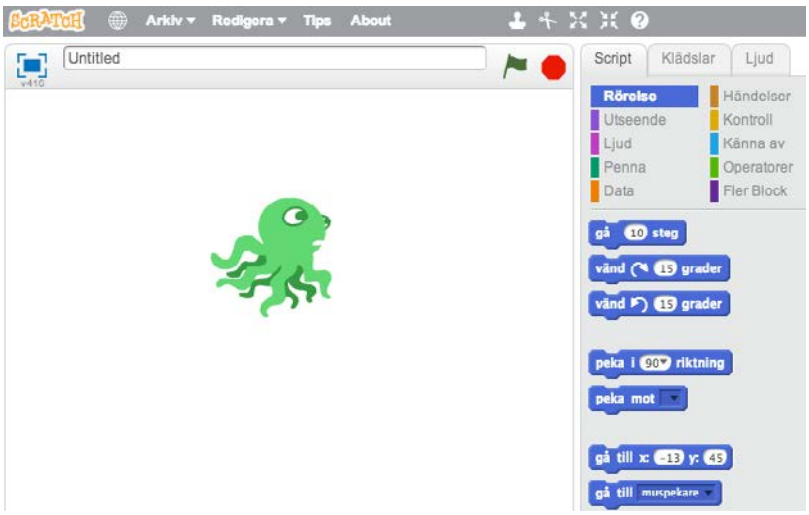
Först och främst, starta ett nytt och tomt Scratchprojekt. Som vanligt laddas katten Scratch in, men vi tar bort honom för nu ska vi använda andra figurer. Högerklicka på katten i sprite-listan och välj ”Radera”

Nu är det dags att ladda in en annan sprite. Scratch kommer med en massa förinstallerade figurer, och för att välja bland dem klickar du på den lilla figuren bredvid ”Ny sprite”.

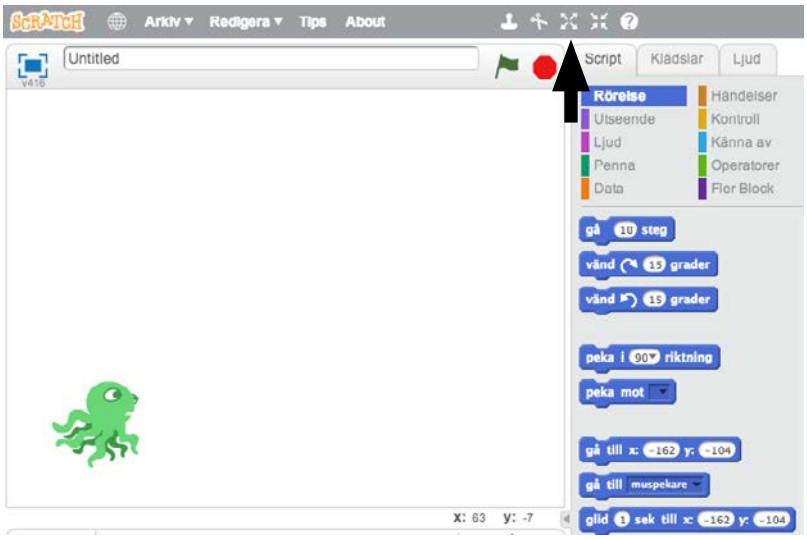
Nu får du upp en lång lista med alla inbyggda sprites. Här finns djur, människor och prylar att välja bland, men vi ska göra ett under-vattenspel, så scrolla ner en bit tills du ser den gröna bläckfisken och dubbelklicka på den för att ladda den.



Bläckfisken laddas nu och lägger sig mitt på scenen. Den är ganska stor, och för att få bättre plats med annat på scenen behöver vi förminska den. I Scratch finns två verktyg högst upp i menyraden för att förstora och förminska, de ser ut som fyra pilar som pekar åt olika håll. Den högra av dem är fyra pilar som pekar inåt, och det är ”förminska”.



Klicka först på den ikonen för att välja verktyget ”förminska” och sen på bläckfisken flera gånger tills den är lagom stor, och dra sen ner den till botten av scenen.

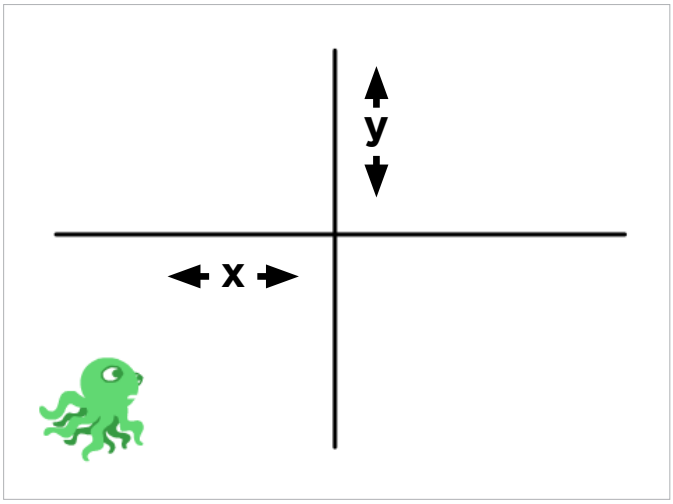


Nu ska vi se till att bläckfisken kan röra på sig, men i stället för att låta den springa fram och tillbaka på skärmen av sig själv ska vi använda piltangenterna för att styra den själva.

Koordinater

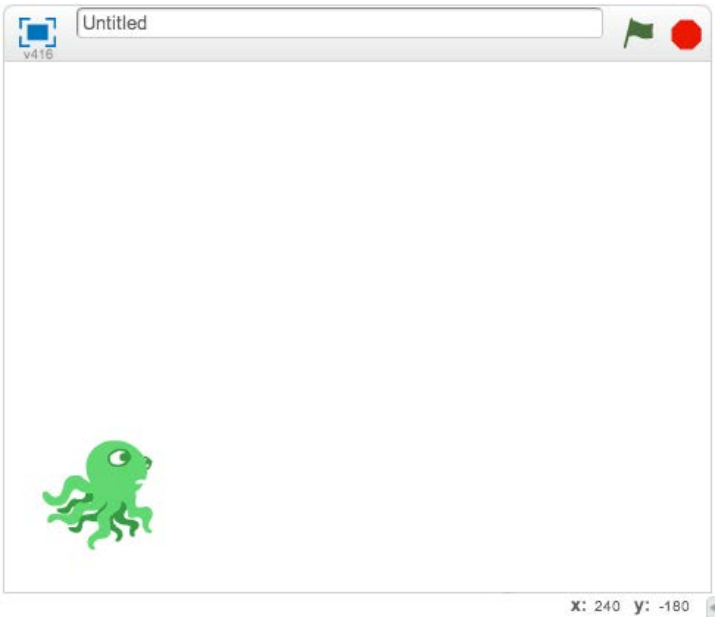
Den här gången ska vi flytta på bläckfisken på ett annat sätt än vi flyttade katten tidigare. I stället för att använda blocket ”gå 10 steg” ska vi använda blocket ”ändra x med 10”. Men först måste vi prata lite om koordinater.

På scenen kan vi flytta på saker genom att ändra eller sätta deras x-värde och y-värde. Vad är då detta? Jo, alla sprites på scenen har en position som anges i koordinater, x och y. En sprites x-koordinat representerar den horisontella position som den befinner sig på, och y-koordinaten anger den vertikala positionen.

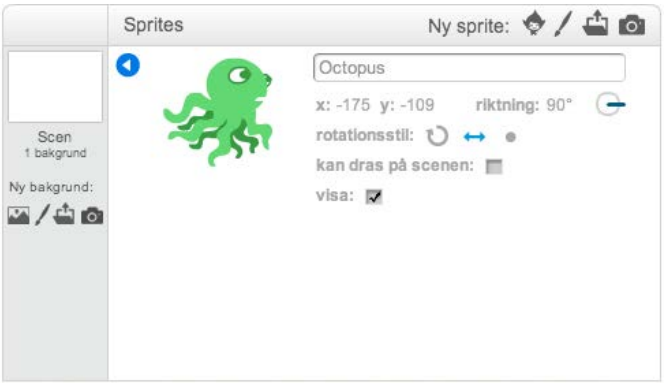


På detta sätt är hela scenen uppstyckad i tusentals pixlar, som alla kan anges med ett x- och ett y-värde. Mittpunkten på scenen är nollpunkt för både x och y, och när saker ligger till höger om mitten har de ett positivt x-värde medan allt till vänster om mitten har ett negativt x-värde. På samma sätt är allt ovanför mitten på scenen på positivt y-värde och allt under mittlinjen har negativa y-värden.

Genom att berätta för Scratch vilka x- och y-värden vi vill att en sprite ska ha kan vi alltså flytta på en figur. Nu när vi har lagt bläckfisken längst ner till vänster har den både negativt x-värde och y-värde. Du kan peka på den med musen för att se ungefär vilka x- och y-positioner den befinner sig på – koordinaterna visas alltid nere till höger under scenkanten.



Tänk på att det alltid är muspekarens x- och y-värde som visas där, inte var en viss sprite befinner sig! För att ta reda på exakt vilka koordinater en sprite befinner sig på behöver vi gå in på inställningarna för den spriten genom att klicka på den lilla ”i”-knappen bredvid bilden i sprite-listan, där vi tidigare ställde in rotationsstilen för en sprite.



Där ser vi vilken position en sprite har vid just detta tillfälle. Just nu spelar det ingen roll exakt var bläckfisken ligger, så stäng inforutan igen genom att klicka på den blå pilen i vänstra hörnet.

Så genom att ändra x- och y-värdena för en sprite kan vi flytta på den. Det är precis det vi ska göra nu, men vi behöver skapa lite struktur för vår kod först, på samma sätt som vi gjort tidigare. Börja med att gå till scriptkategorin ”händelser” dra in ett startblock, ”När grön flagga klickas på” till bläckfiskens script-yta. Gå sen till kategorin ”kontroll” och koppla på ett loop-block, nämligen ”för alltid”. Inuti loopen lägger vi ett villkorsblock, nämligen ”om – då”.

Nu har vi starten på ett program, men vi ska bygga vidare. Till att börja med ska vi använda ett nytt block för att känna av om en piltangent är nedtryckt. Gå till kategorin ”känna av” och dra in blocket ”tangent [mellanslag] nedtryckt?” in i ”om – då”-blocket. Klicka sen på den lilla svarta pilen bredvid ordet ”mellanslag” i blocket och välj ”högerpil”.

Nu har vi skapat ett program som körs ”för alltid” och som känner av ifall tangenten ”högerpil” på tangentbordet är nedtryckt, och använder ett villkor för att i sådana fall göra någonting. Nu är det dags att flytta på bläckfisken!

Flytta på en sprite genom att ändra koordinater

Först och främst vill vi att bläckfisken ska vända sig åt rätt håll när vi flyttar på honom, så vi behöver ett block som pekar spriten. Under kategorin ”rörelse” finns blocket ”peka i [90] riktning”. Dra in det och lägg det inuti villkorsblocket. Om du klickar på den lilla svarta pilen i det här blocket så ser du att du kan välja olika riktningar som anges i grader, men eftersom höger (90 grader) är förvalt kan vi låta det vara just nu. Det här blocket vänder ju bläckfisken i den riktning vi vill, och vi använde det förut för att vända katten åt rätt håll innan vi använde ”gå [10] steg” för att flytta på katten. Nu ska vi ju använda x-koordinater i stället, så egentligen behöver vi inte vända på bläckfisken för att flytta den åt rätt håll, men det ser konstigt ut om bläckfisken inte vänder sig, för då simmar han baklänges ibland!

Precis som förra gången kommer dock spriten att hamna upp-ochned om vi vänder honom utan att ändra rotationsstil. Tidigare gick vi in i spritens inställningar för att ändra rotationsstil, men det finns ett sätt att göra det med block också, och det ska vi göra nu. Under kategorin ”rörelse” finns blocket ”sätt rotationsstil [vänster–höger]” som gör precis det vi vill. Det här blocket räcker det om vi kör en gång, så dra in det och släpp det precis under startblocket, ovanför loopen.

Under kategorin ”rörelse” finns också blocket ”ändra x med [10]”. Dra in det och lägg det inuti villkorsblocket, men ändra sedan siffran tio till en trea. Det här flyttar alltså på bläckfisken genom att öka x-koordinaten med tre i loopen, vilket gör att bläckfiskens x-värde blir högre och alltså att den rör sig till höger i koordinatsystemet.

Nu kan du testa genom att klicka på den gröna flaggan. Inget händer genast, men om du trycker ner högerpilen på ditt tangentbord så börjar bläckfisken simma till höger. För att han ska kunna simma åt andra hållet behöver vi skapa ett till villkor och lägga in samma block en gång till, men ändra lite saker.

Börja med att ta villkorsblocket ”om – då” från kategorin ”kontroll” och släpp det under vårt första villkorsblock, men fortfarande inne i loopen. Gå sen till ”känna av” och ta blocket ”tangent [mellanslag] nedtryckt” igen, släpp det i villkoret men ändra sen ”mellanslag” till ”vänsterpil”.

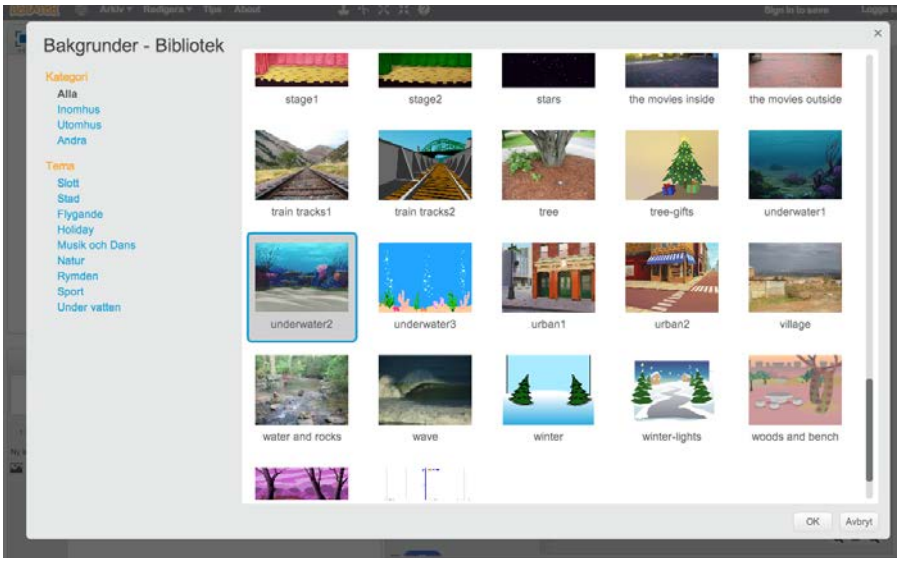
Gå sen till kategorin ”rörelse” och ta blocket ”peka i [90] riktning” och släpp det inne i vårt nya villkor, men ändra nu ”90” till ”-90” (vänster). Ta sen blocket ”ändra x med [10]” och släpp det under peka-blocket. Nu blir det lite klurigt, för om vi ökade värdet på x-koordinaten i förra villkoret för att flytta bläckfisken till höger måste vi nu minska värdet på x-koordinaten för att flytta bläckfisken till vänster. Ändra siffran tio till ”-3”, alltså minus tre. Det här kommer att dra tre ifrån x-värdet och alltså flytta spriten till vänster i koordinat-systemet.

Testa genom att klicka på den gröna flaggan igen. Nu kan du låta bläckfisken simma fram och tillbaka genom att trycka på vänster- och högerpilarna på ditt tangentbord.

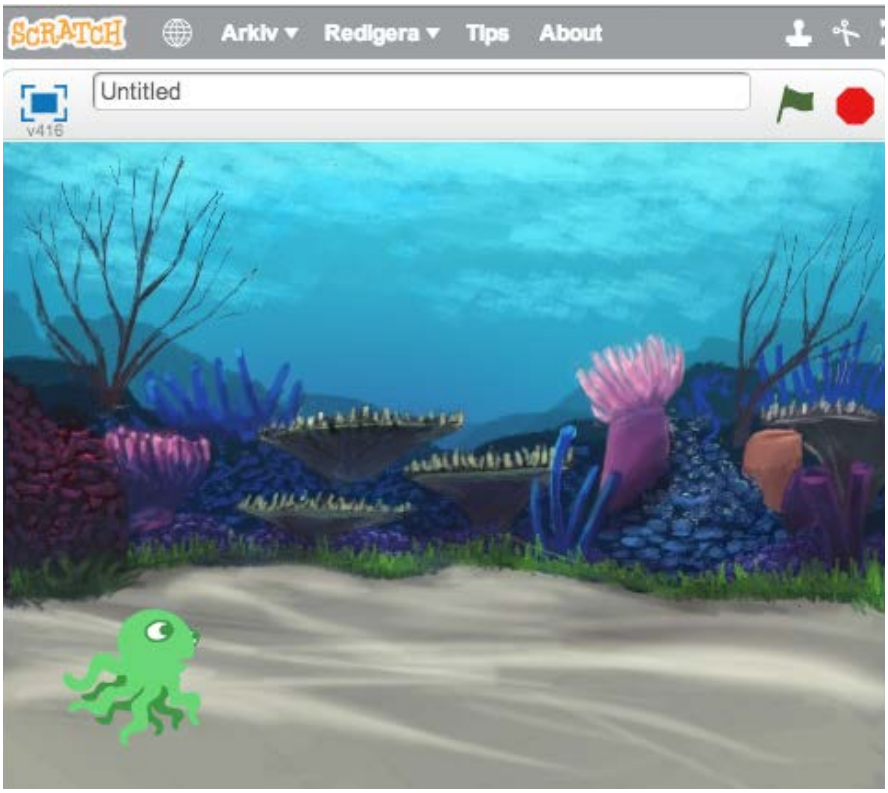
Ladda bakgrund

Vi gör ett undervattensspel, så låt oss hämta en passande bakgrund. Till vänster om sprite-listan, under ikonen som visar scenen, finns en liten ikon som ser ut som en tavla. Klicka på den för att öppna det inbyggda biblioteket med bakgrunder.

I den långa listan med inbyggda bakgrundsbilder finns, en bra bit ner, en som heter ”underwater2” som är perfekt för det här spelet. Dubbelklicka på den för att ladda in den och lägga den som bakgrund till scenen.



Nu är bakgrunden laddad, och vår bläckfisk ligger ovanpå den. Se till att han ligger rätt i höjd, så att det ser ut som att han simmar på havsbotten.



Animera bläckfisken

När bläckfisken simmar fram och tillbaka så glider han liksom över havsbotten. Just denna sprite kommer med två ”kostymer”. Vi pratade om klädslar och kostymer redan i förra kursen, och även denna gång kan vi växla mellan de två kostymerna genom att skapa ett separat script för det.

Klicka på bläckfisken i sprite-listan för att välja den, så ser du scriptet för att röra bläckfisken igen. Gå nu till kategorin ”händelser” och dra in ett nytt ”när grön flagga klickas på”-block och lägg det bredvid vårt förra program i scriptytan. Gå sen till ”kontroll” och ta en loop – ”för alltid” och lägg den under startblocket. Gå sen till ”utseende” och ta blocket ”nästa klädsel” och lägg det inne i loopen. Till slut går du tillbaka till ”kontroll” och väljer ”vänta [1] sekunder” och lägger det under ”nästa klädsel”, inne i loopen. Ändra ettan så att det står ”vänta 0.2 sekunder” i stället.

Testa genom att klicka på den gröna flaggan. Nu ser det ut som om bläckfisken simmar i vattnet eftersom den växlar mellan de två klädslarna. Du kan fortfarande styra bläckfisken genom att trycka på höger- eller vänsterpiltangenterna.

Kloner – ett sätt att skapa många likadana figurer på scenen

Nu ska vi se till att det finns något att göra i spelet utöver att styra bläckfisken. Vi ska ha stjärnor som dyker upp på havsbotten som vi ska försöka fånga för att få poäng, och vi ska ha farliga hajar som kommer simmande och elaka krabbor som dyker upp och försöker knipsa oss. Alla dessa objekt ska vi bara programmera en gång, och sen låta Scratch skapa ett antal av dem åt oss genom att använda oss av ”kloner”.

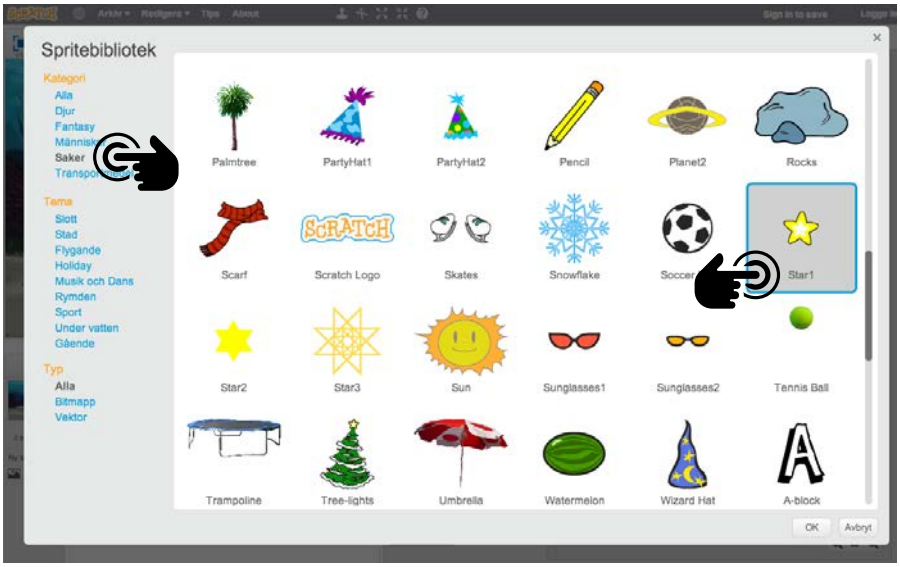
Kloner kan sägas vara Scratchs introduktion till objektorienterad programmering. Poängen med OOP, som det förkortas, är att du bara behöver skriva kod en gång, men kan återanvända den i olika ”instanser”, eller kloner som det kallas i Scratch. I det här fallet ska vi alltså programmera beteendet för en stjärna, en haj och en krabba, och sedan ska vi använda vårt scenscript för att dra igång olika antal av dessa objekt. Eftersom Scratch har särskilda block för att hantera kloner är det inte så komplicerat, men det kan vara lite klurigt att förstå hur det hänger ihop från början.

Tänk dig att vi skulle vilja göra ett spel med en cowboy som skjuter en pistol. Vi skulle då ha en sprite för själva cowboyen, skapa script för den som gör så att han kan gå fram och tillbaka och kanske skjuta. Men själva kulorna från pistolen, hur skulle vi programmera dem om vi vill att man ska kunna skjuta många skott i taget? I stället för att ladda in flera likadana sprites och lägga exakt samma script i allihop kan vi med hjälp av kloner ladda en skott-sprite, programmera den en gång och sen skapa hur många sådana skott vi vill, genom att säga åt Scratch att sätta igång nya kloner av den.

Varje klon kan bara göra precis det som vi programmerat in i den enda sprite vi laddat. Om vi vill att olika skott ska bete sig på olika sätt får vi antingen lägga in programmeringen för att hantera det, eller ladda fler sprites. I vårt undervattenspel ska vi bara använda en stjärna, en haj och en krabba. Kloningen sköter resten.

Ladda sprite som ska bli klonad

Vi börjar med stjärnan. I spelet ska vi se till att det dyker upp stjärnor på olika ställen på havsbotten. Målet med spelet är att fånga så många som möjligt av dem utan att bli uppätna av hajen eller knipsade av krabban. Ladda in en ny sprite genom att klicka på den lilla gubben ovanför sprite-listan.



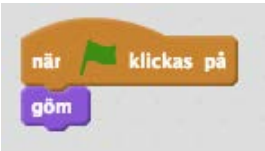
I det stora sprite-biblioteket kan du snabbare hitta rätt genom att välja en kategori till vänster. Stjärnan hittar vi i kategorin ”saker”, så klicka på ”saker” till vänster och scrolla sen ner för att hitta ”Star1”. Dubbelklicka på den för att ladda in den.

Stjärnan laddas in och lägger sig mitt på scenen. Där vill vi ju inte ha den, för bläckfisken simmar bara på botten. Vi behöver alltså lägga den någonstans på botten, där vi kan nå den sen. Dra ner stjärnan med musen så att den ligger någonstans på botten. Det spelar ingen roll var i sidled den ligger, för vi ska snart använda programmering för att flytta på den. Först av allt behöver vi se till att den är osynlig när spelet startar, och sen behöver vi skapa själva programmet som ska köras när stjärnan startar som en klon.

Göm en sprite

Ibland i spel är det praktiskt att kunna bestämma när saker ska visas på scenen och inte. I Scratch är det väldigt enkelt, för vi har två block under ”utseende” som heter ”visa” och ”göm” och de gör precis vad de säger. De flyttar alltså faktiskt inte på spriten, utan gör den bara osynlig eller synlig.

I vårt spel ska stjärnan inte visas från början, så vi tar ett start-block, ”när grön flagga klickas på” under ”händelser” och drar ut det till stjärnans script-yta. Klicka sen på ”utseende” och dra in blocket ”göm” och lägg det under startblocket.



Klicka på den gröna flaggan nu, så ser du att stjärnan genast blir osynlig. Om du av någon anledning behöver visa en osynlig sprite igen så kan du alltid gå in i inställningarna för den spriten och klicka på rutan ”visa”. På samma sätt kan du gömma en sprite manuellt.

Nu när vi skapat det lilla programmet som gömmer stjärnan när spelet börjar är det dags att sätta igång med scriptet som slumpmässigt ska bestämma var stjärnan ska hamna, visar den i ett antal sekunder och sen gömmer den igen. Eftersom vi vill använda kloner måste vi använda ett alldeles särskilt startblock, nämligen ”när jag startar som klon”, och det ligger märkligt nog inte under ”händelser”, utan under ”kontroll”. Leta upp det och dra in det till stjärnans scriptyta, bredvid vårt andra program.

Det här startblocket körs alltså bara när vi säger åt Scratch att vi vill ha en ny klon av denna sprite, vilket vi kommer att göra lite senare, när vi programmerat klart stjärnans beteende.

Slumptal

För att spelet ska bli omväxlande kan vi ju inte låta stjärnan ligga på samma ställe varje gång den dyker upp, då skulle det bli alldeles för lätt för vår bläckfisk att hänga där och vänta på att den kommer. Därför behöver vi slumpa positionen som stjärnan ligger på innan den visas. I Scratch finns en väldigt enkel funktion för att skapa ett slumptal, och den kan vi använda för att slumpa fram stjärnans x-värde, det vill säga var i sidled stjärnan ska dyka upp. Den måste ju alltid ligga på botten, annars kan inte vår bläckfisk nå den – så y-värdet behöver vi aldrig ändra på.

I Scratch är scenen exakt 480 pixlar bred, och eftersom mitten av scenen har x-värdet noll är högra kanten av scenen x 240 och vänstra kanten av scenen x -240. För att stjärnan ska hamna på olika ställen i sidled kan vi alltså slumpa fram ett x-värde som ligger mellan -240 och 240, men då skulle vi riskera att en del av stjärnan hamnar utanför scenen, så vi väljer att använda -220 och 220 i stället.

Under kategorin ”rörelse” finns ett annat block som hanterar x-värde för en sprite, nämligen ”sätt x till [0]”. Till skillnad från det förra x-blocket vi använde så bestämmer det här blocket exakt x-värde och struntar i vilket x-värde spriten för tillfället har. Det funkar på samma sätt som våra variabelblock vi använde i förra kursen. Ta tag i blocket och dra in det under ”när jag startar som klon”.

Nu ska vi så ta fram vårt slumptal, och det gör vi med ett block som finns under ”operatorer”. Blocket heter ”slumptal [1] till [10]”, och det kan vi dra in och släppa i rutan i blocket där det nu står ”sätt x till [0]” så att det står ”sätt x till slumptal 1 till 10”. Se till att släppa det på rätt ställe så att det verkligen hamnar i rutan och inte utanför.

I stället för ett och tio ska vi ju ha -220 och 220, så klicka i rutorna och skriv in våra värden, så att det står ”sätt x till slumptal -220 till 220”.

Nu har vi med hjälp av det här blocket flyttat på stjärnan, fast det kommer inte att synas direkt eftersom stjärnan är gömd när vi börjar. Vi behöver alltså visa stjärnan efter att vi flyttat på den, så gå till kategorin ”utseende” och dra in blocket ”visa”.

Nu vill vi att stjärnan ska ligga där i några sekunder och sen försvinna igen. Vi kan använda slumpen igen för att bestämma om stjärnan ska ligga där i två sekunder eller fem. Gå till ”kontroll” och dra in blocket ”vänta [1] sekunder”. Gå sen tillbaka till operatorer och dra in blocket ”slumptal [1] till [10]” och släpp det i rutan där det nu står en etta. Ändra de två slumptalen till två och fem, så att stjärnan väntar mellan två och fem sekunder varje gång.

Efter att stjärnan legat där i några sekunder ska den försvinna igen, så gå till ”utseende” och dra in blocket ”göm”.

Sist av allt behöver vi radera klonen, för efter att stjärnan visats och gömts är den färdig och ska tas bort. Att radera en klon innebär bara att just den instans av spriten som vi startat försvinner, inte att själva spriten och all dess kod försvinner. Gå till ”kontroll” och dra in blocket ”radera klonen” och lägg det sist.

Vårt program väntar alltså på att vi ska säga till att vi vill starta en klon av stjärnan, och då flyttar den på stjärnan, visar den, väntar en stund och gömmer den igen. Sist av allt raderar klonen sig själv, eftersom den är färdig.

Skapa en klon

Nu ska vi använda vår stjärna genom att skapa kloner av den. Vi tycker att det är prydligt att lägga de program som är övergripande för hela spelet i scenens scriptyta, så klicka på scenen till vänster om sprite-listan. Scenen har inga program än, så det är tomt i scriptytan till att börja med.

Vi vill att stjärnorna ska dyka upp vid olika tillfällen, så det är dags för slumpen igen. Börja med att dra in startblocket ”När grön flagga klickas på” och koppla på loopen ”för alltid”. Gå sen till ”kontroll” och lägg blocket ”vänta [1] sekunder” inne i loopen. Gå till ”operatorer” och dra in ”slumptal [1] till [10] sekunder” i rutan där det står en etta. Ändra värdena för slumptalen till en etta och en femma, så att det står ”slumptal 1 till 5 sekunder”.

Nu ska vi skapa själva klonen. Under kategorin ”kontroll” finns blocket ”skapa klon av”. Dra in det och lägg det under vänta-blocket. Klicka på den lilla svarta pilen för att välja ”Star1” så att det står ”starta klon av Star1”. Nu kommer vi att under hela spelets gång skapa nya kloner av stjärnan efter ett slumpmässigt intervall.

Testa nu genom att klicka på den gröna flaggan. Bläckfisken simmar och då och då dyker en stjärna upp på botten, men försvinner igen efter ett tag. Om du tittar en stund kommer du att få se att det kan bli flera stjärnor samtidigt! Vi har ju sagt att en ny klon av stjärnan ska skapas varje 1–5 sekunder, så med lite tur kan vi få flera stjärnor att dyka upp innan de gamla hunnit försvinna.

Kollisioner – rör de vid varandra?

Nu när vi har stjärnor som dyker upp på havsbotten är det dags att börja räkna poäng. Varje gång bläckfisken lyckas ta en stjärna ska stjärnan försvinna, man ska få en poäng och så kan vi spela upp ett litet ljud så att man hör att man lyckas.

Redan i förra kursen räknade vi poäng med variabler, så vi gör på liknande sätt denna gång. Gå till kategorin ”data” och klicka på ”skapa en variabel”. Döp den till ”Poäng” och tryck på ”OK”. Nu dyker vår poängtavla upp på scenen och vi får variabelblock att använda oss av.

Klicka på stjärnan i sprite-listan så att du ser stjärnans program igen. Nu behöver vi skapa ett program till här som håller koll på om stjärnan och bläckfisken rör vid varandra, och även detta program körs ju när vi klonat stjärnan, så gå till ”kontroll” och dra in ett nytt startblock ”när jag startar som klon” och lägg den bredvid dina andra program för stjärnan. Dra sen in en för alltid-loop och ett om-då-villkor i loopen.

Scratch kan hålla koll på om sprites rör vid varandra, vilket är praktiskt i väldigt många olika slags spel. I det här fallet vill vi använda den funktionen för att se om stjärnan rör vid bläckfisken, så gå till ”känna av” och dra in det diamantformade blocket ”rör” och släpp det i om-då-blocket. Klicka på den lilla svarta pilen och välj ”octopus” så att vi kollar om ”jag” (alltså klonen) rör vid bläckfisken och inget annat.

Vi ska få en poäng när bläckfisken rör vid stjärnan, så gå till ”data” och dra in blocket ”ändra [poäng] med 1”. Gå sen till kategorin ”ljud” och dra in blocket ”spela ljudet [pop]”, så att det låter lite när vi lyckas ta en stjärna. När vi fått poäng och spelat ljud så ska stjärnan försvinna, så gå till ”kontroll” och dra in blocket ”radera klonen”.

Vårt nya program körs alltså när stjärnan startas som en klon, det loopar och kollar om klonen rör vid bläckfisken, och om det händer får vi en poäng, spelar ett ljud och raderar klonen.

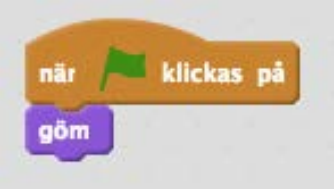
Testa nu spelet och se om du hinner simma ifatt stjärnorna och att du får poäng när du lyckas. Ibland hör du ett ”pop” och får ett poäng utan att du hinner se någon stjärna, och det är när det slumpar sig så att en stjärna dyker upp på precis den plats där bläckfisken råkar befinna sig för tillfället. Gratispoäng!

Mer kloner – hajen

Nu kan vi ju spela spelet, men för att det ska bli lite svårare ska vi lägga till en farlig haj och en krabba. Även dessa ska vi göra som kloner, men de ska bete sig på lite olika sätt. Vi börjar med hajen.

Ladda in en ny sprite genom att klicka på den lilla gubben ovanför sprite-listan igen. Hajen som vi ska använda heter ”Shark” och ligger nästa längst ner i kategorin ”Djur”.

Nu ska vi bygga programmet för hajen. Vi vill ju inte att hajen ska synas när spelet börjar, så ta en ”när grön flagga klickas på” och ett ”göm”.



Gå sen till ”kontroll” och dra in ett ”när jag startar som klon”, för även hajens program ska främst köras när vi klonar den. Hajen ska få simma snett över skärmen, från det övre vänstra hörnet och ner till höger, fast lite olika varje gång. Det kan vi uppnå genom att slumpmässigt rotera hajen lite innan den visas och börjar åka.

Först av allt när klonen skapas ska vi placera hajen på sin startposition i övre vänstra hörnet. Hajen är ganska stor, så den behöver programmeras att flytta en bit utanför scenen för att dyka upp på ett snyggt sätt. Gå till ”rörelse” och dra in ett block som heter ”gå till x: y:”. Vi mätte upp hajen och kom fram till att en bra startposition är x -270, y 235. Det här kan du ändra senare om du vill, men skriv in ”-270” i rutan för x-värde och ”235” i rutan för y-värde nu.
För att hajen ska simma på lite olika ställen varje gång ska vi nu rotera den med lite slumptal. Under ”utseende” finns blocket ”peka i [90] riktning”. Dra in det och släpp det under ”gå till”-blocket. Gå sedan till ”operatorer” och dra in slumptalsblocket i rutan för riktningsvärde. Mellan 110 och 180 grader verkar vara lagom för hajen, så skriv ”110” i den första rutan och ”180” i den andra.

Efter att vi pekat hajen i rätt riktning är det dags att visa den och sätta igång att flytta på den. Dra in blocket ”visa” som ligger under kategorin ”utseende” och släpp det efter peka-blocket.

För att hajen ska simma fram av sig själv behöver vi nu en för alltid-loop. Gå till ”kontroll” och dra in loopen och släpp den under ”visa”. Nu kan vi använda det gamla fina blocket ”Gå [10] steg” för att flytta på hajen – det ligger under ”rörelse”. Lägg det inne i loopen och ändra siffran tio till en fyra, så rör sig hajen lagom fort.

Till sist behöver vi ta bort hajen när den simmat fram över underkanten, så nu behöver vi använda ett villkor igen för att kolla om hajen simmat färdigt. Vi kan använda hajens y-värde för att kolla var den befinner sig i höjdled, och ett villkorsblock för att göra någonting när den kommit fram. Gå till ”kontroll” och ta ett ”om – då” och släpp det i loopen, under gå-blocket. Det vi vill kolla är om hajens y-läge är under scenens kant. Scenens mitt är ju nollpunkten, och när vi rör oss nedåt på scenen blir det negativa y-värden. Nederkanten av scenen är y -180, men hajen ska få en chans att simma lite till så att den helt försvinner ur bilden, så vi använder -230 i stället.

För att jämföra två värden behöver vi gå till kategorin ”operatorer” och hitta det diamantformade blocket som kollar om ett värde är mindre än ett annat.

Släpp det blocket i den diamantformade rutan i villkorsblocket. Vi kan nu kontrollera om ett värde är mindre än ett annat, och vi vill ju veta om hajens y-värde är mindre än -220 för att se om den simmat över underkanten. Att hämta hajens y-läge kan vi göra under kategorin ”rörelse”, med blocket ”y-läge”. Dra in blocket i den ena vita rutan i den gröna operatorn, klicka sen på den andra vita rutan och skriv in ”-230”.

Det vi vill ska hända om hajen simmat över kanten är att den ska försvinna helt, så gå till ”kontroll” och dra in ett ”radera klonen”-block.

Det här programmet fick många olika delar, men om vi läser det uppifrån och ned är det inte så jobbigt. När klonen startas, lägg hajen i det övre vänstra hörnet av scenen, peka den i lite slumpad riktning och visa den. Sätt sedan igång en loop så att hajen simmar framåt i den riktning den blivit pekad, känn av när den simmat förbi kanten och radera klonen då.

Nu behöver vi ett script som startar haj-klonerna, och det lägger vi precis som scriptet som visar stjärnorna i scenens script. Klicka alltså på den lilla undervattensbilden till vänster om sprite-listan för att visa scenens script. Nu ska vi skapa ett script som är nästan identiskt med det vi använde för att starta kloner av stjärnan, men

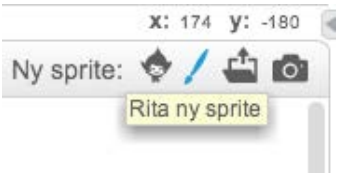
hajen behöver inte dyka upp lika ofta, så vi kan använda andra tal i slumpgeneratorn. Mellan fem och tio sekunders paus mellan hajarna känns lagom.

Vi har fortfarande inte sett till att något händer när hajen lyckas äta upp bläckfisken. Klicka på hajen igen i sprite-listan så sätter vi igång.

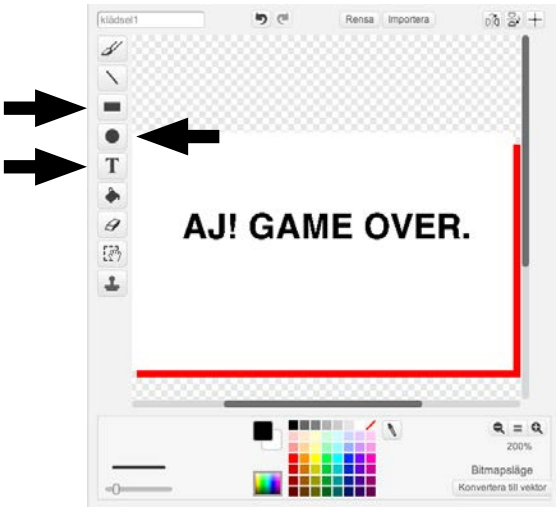
Det här är ju nästan exakt samma program som vi skapade för stjärnan, men vi vill spela ett annat ljud och dessutom att spelet ska ta slut när hajen rör bläckfisken. Först och främst behöver vi ett nytt ”När jag startar som klon”-block, som ligger under ”kontroll”. Ta sedan en ”för alltid”-loop och inuti den ett ”om-då”-villkor. Under ”kontroll” har vi blocket ”rör” igen, lägg det i villkoret och välj ”Octopus” så att vi känner av om hajen rör vid bläckfisken. Nu kan vi få saker att hända om en haj lyckas få tag i en bläckfisk, men vi behöver ju också meddela spelaren att det är game over. Det här är ett bra tillfälle att titta på hur man gör egen grafik i Scratch.

Gör en egen sprite

Vi vill göra en skylt som det står ”GAME OVER” på. Eftersom den här skylten ska dyka upp först när en haj tagit bläckfisken kan vi göra skylten som en separat sprite som vi gömmer när spelet börjar och sedan visar när spelet är slut. För att rita en helt egen sprite kan vi använda Scratch inbyggda ritprogram, klicka på den lilla penseln över sprite-listan för att skapa en ny, tom sprite:



Vi får en ny, tom sprite som heter ”Sprite1” eller liknande, och vi kommer direkt in i det inbyggda ritprogrammet. Nu kan vi skapa en enkel skylt genom att använda verktyget för rektanglar och textverktyget. Rita ett par rektanglar och använd fyllningshinken för att fylla dem så att du får en vit skylt. Klicka sen på textverktyget och skriv in ”GAME OVER” på skylten.



Den här skylten ska vi bara visa när spelet är slut, men medan vi ritar den så visas den direkt på scenen. Nu ska vi programmera den. Klicka därför på fliken ”Script” för denna sprite och dra in ett ”när grön flagga klickas på” och ett ”göm”. Nu kommer ju skylten att försvinna så fort vi startar spelet, vilket är precis vad vi vill. Men för att vi ska kunna programmera in att skylten ska visas så fort hajen tagit bläckfisken behöver vi ju kunna få information från vårt andra program. Det gör vi i Scratch med hjälp av något som kallas ”meddelanden”.

Meddelanden – låt programmen prata med varandra

Hittills har vi skapat ett antal olika program som ”bott” i en sprite eller scenen. De har skött sig själva, och inte haft något med varandra att göra mer än att känna av om en sprite rör vid en annan. Men om vi nu vill att något ska hända med en sprite när något annat händer i vårt spel behöver vi använda meddelanden, som är ett sätt att låta olika program prata med varandra, oavsett var de bor.

Ett meddelande är helt enkelt ett litet tillrop mellan två program. Vi kan skicka ett meddelande och vi kan lyssna på ett meddelande. Alla meddelanden skickas genom hela Scratch, så om du till exempel skickar ett meddelande från bläckfisken så behöver du inte berätta vem du skickar meddelandet till – det räcker att börja lyssna där du vill plocka upp det. Du kan skapa många olika meddelanden och använda dem till olika saker.

Nu ska vi skapa ett meddelande som skickas från hajens script när den lyckats fånga bläckfisken, och som tas emot av skyltens script. Klicka på hajen i sprite-listan för att gå tillbaka till hajens program. Här finns vårt script som vi gjorde för att känna av om hajen rör vid bläckfisken. Gå nu till kategorin ”kontroll” och hitta blocket ”skicka [message1]”, näst längst ner. Dra in det och släpp det i villkoret.

Scratch har redan skapat ett meddelande åt oss, men vi tar och gör ett eget. Klicka på den lilla svarta pilen i blocket du just drog in och välj ”nytt meddelande”. I rutan som poppar upp kan du skriva vad du vill kalla ditt nya meddelande. Vi kallar det ”game over” den här gången så skriv det i rutan. När du sparat meddelandet ser du att blocket nu valt ditt nya meddelande i stället, så det står ”skicka [game over]”. När hajen fångar bläckfisken kommer nu meddelandet ”game over” att skickas över hela Scratch, och alla andra script kan ”lyssna” på det.

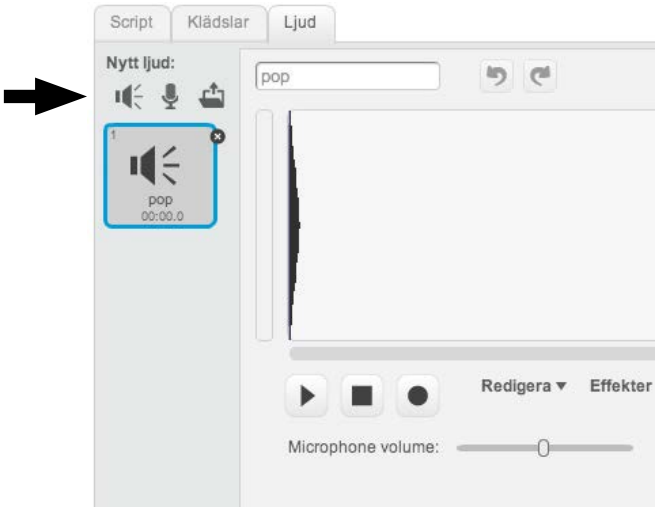
Efter att vi skickat meddelandet ska vi också radera klonen, så gå till ”kontroll” och dra in ”radera klonen”.

Nu ska vi se till att vår sprite med skylten lyssnar på meddelandet och visar sig när det tas emot. Klicka på skylten i sprite-listan så att du ser skyltens program igen. Under kategorin ”Händelser” finns startblocket ”När jag tar emot [game over]”. Det senaste meddelandet som skapats är förvalt, men om det står något annat där

kan du klicka på den lilla svarta pilen och välja ditt meddelande. Dra in blocket och börja på ett nytt script. Det vi ska göra när vi tar emot meddelandet är att visa skylten och sen avsluta spelet, så gå till ”utseende” och dra in ett ”visa”.

Ladda in ljud

Nu skulle vi vilja spela ett riktigt läskigt ljud om vi blir uppätna, för då är spelet slut. Klicka på fliken ”ljud” som ligger ovanför scriptlistan så ser du att vi bara har ett ljud till hajen just nu, nämligen ”pop” som vi använde när bläckfisken fångar en stjärna. Vi laddar in ett nytt ljud från de som kommer inbyggda i Scratch genom att klicka på den lilla ikonen som ser ut som en högtalare.



Även här kan man bläddra mellan ljuden genom att klicka på kategorierna till vänster. Under kategorin ”Elektronisk” ligger ett ljud som passar bra, det heter ”screech”. Dubbelklicka på det för att ladda in det. Klicka sen på ”Script”-fliken så att du ser skyltens program igen.

Det finns ett par olika block som har med ljud att göra under kategorin ”Ljud”. Det vi använt förut heter ”spela ljudet” men det finns ett block under det som heter ”spela ljudet tills färdigt”, och det ska vi använda denna gång. Dra in det och släpp det under ”visa”. Där ska ljudet ”screech” redan vara valt, men dubbelkolla för säkerhets skull.

Nu kommer det läskiga ljudet att spelas när spelet tar slut.

Stoppa spelet

För att spelet verkligen ska ta slut behöver vi också se till att stoppa allt efter att det blivit game over. Gå till ”kontroll”. Leta upp blocket ”stoppa [alla]” och dra in det efter ljudblocket.

”Stoppa alla” gör just det – samma sak som om du hade tryckt på stoppknappen högst upp. När spelet är slut måste man starta om det genom att trycka på den gröna flaggan igen.

Initscript – när spelet börjar

Nu börjar spelet ta form, men vid det här laget har du kanske upptäckt att vi inte nollställer poängen varje gång spelet startas. Det här pratade vi om även i förra kursen, men vi behöver ett ”initscript”, något som nollställer det som behövs varje gång spelet startas. Det här scriptet kan vi lägga i scenens script, så klicka på scenens ikon bredvid sprite-listan igen.

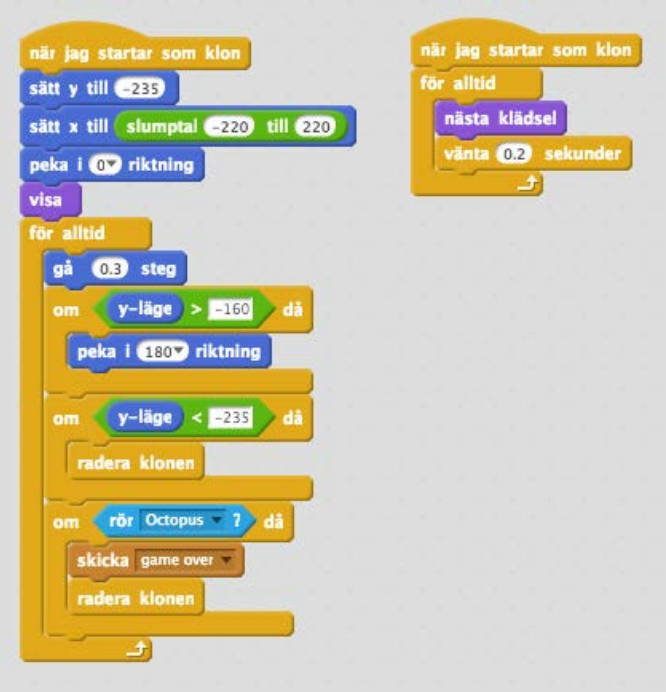
Dra in ett nytt startblock, ”när grön flagga klickas på”, och gå sen till ”data” och dra in ett ”sätt [poäng] till [0]”. Nu har vi sett till att poängen nollställs i början av varje spelomgång.

Ännu en klon – krabban

Hajen är farlig, men vi tänkte att vi skulle lägga in en krabba också som kommer upp från botten av havet då och då och försöker knipsa vår bläckfisk. Nästan all programmering för krabban kommer att vara likadan som den för hajen.

Först laddar vi in en ny sprite, krabban heter ”Crab” och ligger under ”djur”. Gå till krabbans script och dra in ett ”när grön flagga klickas på” och lägg till ett ”göm”, så att krabban inte syns när spelet börjar.

Nu ska vi göra scriptet som hanterar en klon av krabban. Titta på bilden så ser du hur vi programmerat och kan göra likadant. Nedanför bilden förklarar vi skillnaderna mot hajens script.



Som du ser har vi två script som körs när krabban startas som klon. Det lilla scriptet animerar bara krabban – den kommer med två klädslar, precis som bläckfisken, och om vi växlar mellan dem så ser det ut som om krabban knipsar med klorna.

Det andra, längre scriptet börjar med att sätta y-värdet så att krabban ligger längst ner på skärmen, och sätta y-värdet med hjälp av slumpen så att krabban dyker upp på olika ställen längs botten. Sedan pekas krabban rakt uppåt (noll graders riktning) men för att den inte ska snurra runt så sätter vi rotationsstilen till ”rotera inte” – då visas spriten rakt upp, oavsett rotation. Sen visar vi krabban.

I loopen flyttar sig krabban långsamt, nämligen 0.3 steg i taget rakt upp (för vi satte ju riktningen till uppåt tidigare), och sedan kommer tre villkor efter varandra. Det första kollar om krabban hunnit simma upp en bit, nämligen till ett y-läge som är mer än -160. Då vänder vi helt om på krabban så att den pekar neråt, och därför börja simma tillbaka mot botten i stället.

Nästa villkor kollar om krabban hunnit simma tillbaka ner till botten av skärmen och raderar klonen när den är färdig. Det sista villkoret känner av om krabban rör vid bläckfisken, skickar ”game over”-meddelandet, som är detsamma som hajen skickar, och raderar klonen.

Vi valde att låta krabban röra sig väldigt långsamt eftersom det blir svårt för bläckfisken att hinna undan annars. Om du ändå tycker att det är för svårt när du testspelar kan du flytta upp bläckfisken på scenen lite, men tänk på att du kanske behöver flytta upp stjärnorna också då.

Glöm inte att du behöver ett script i scenen för att sätta igång krabbans kloner också. Vi satte tiden som slumpas till mellan 15 och 25 sekunder, annars kommer det för många krabbor, men testa gärna med olika värden!

Bakgrundsljud – att loopa ljud

Spelet börjar verkligen ta form, men vi har ett par saker kvar att göra innan det är helt klart. Först och främst skulle vi vilja att det låter som om vi är under havet när vi spelar. För att göra det kan vi lägga ett script i scenen som loopar ett bakgrundsljud.

Klicka på scenen och klicka sen på fliken ”Ljud”. Klicka på den lilla högtalaren för att ladda in ett nytt ljud. Ljudet vi ska ha heter ”bubbles” och ligger under kategorin ”Effekter”.

När du laddat in ljudet, ta och börja på ett nytt script genom att dra in ett ”när grön flagga klickas på”-block. Ta sen en för alltid-loop och i den lägger du ljudblocket ”spela ljudet [bubbles] tills färdigt”. Nu har du gjort en ljudloop. Om du väljer fel block här, och i stället tar det block som bara heter ”spela ljudet [bubbles]” så kommer Scratch att försöka spela ditt bubbelljud om och om igen så snabbt det kan, och det låter inte så bra. Testa!

Spelet är färdigt! Hur många poäng kan du få? Vi fick som mest 23 när vi testade.

Det här spelet finns att spela på http://scratch.mit.edu/projects/21528024/.

Måns Jonasson

Måns Jonasson är varumärkesansvarig på IIS och håller sedan flera år kostnadsfria programmeringskurser för barn och deras föräldrar. Du hittar Måns videokurser på barnhack.se

Foto: Sara Arnald CC-BY ND

Kom igång med Scratch!
Programmera ett spel, steg för steg i Scratch.
IIS internetguide, nr 40. Version 2.0 2016
Måns Jonasson

IIS klimatkompenserar för sina koldioxidutsläpp och stödjer klimatinitiativet ZeroMission.

Författare: Måns Jonasson
Redaktör: Hasse Nilsson
Projektledare: Jessica Bäck
Formgivning: AGoodId
ISBN: 978-91-7611-684-5

I denna guide bidrar Polisens Nationella Bedrägericenter med erfarenheter och tips kring brottsförebyggande åtgärder.

Vi driver internet framåt! IIS arbetar aktivt för positiv tillväxt av internet i Sverige. Det gör vi bland annat via projekt som samtliga driver utvecklingen framåt och gynnar internetanvändandet för alla. Exempel på pågående projekt är:

Bredbandskollen
Sveriges enda oberoende konsumenttjänst för kontroll av bredbandsuppkoppling. Med den kan du på ett enkelt sätt testa din bredbandshastighet. www.bredbandskollen.se

Internetdagarna
Varje höst anordnar vi Internetdagarna som är Sveriges ledande evenemang inom sitt område. Vad som för tio år sedan var ett forum för tekniker har med åren utvecklats till att omfatta samhällsfrågor och utvecklingen av innehållet på internet. www.internetdagarna.se

Internetfonden
Hos Internetfonden kan du ansöka om finansiering för fristående projekt som främjar internetutvecklingen i Sverige. Varje år genomförs två allmänna utlysningar, en i januari och en i augusti. www.internetfonden.se

Internetguider
IIS publicerar kostnadsfria guider inom en rad internetrelaterade ämnesområden, som webb, pdf eller i tryckt format och ibland med extramaterial.

Internetstatistik
Vi tar fram den årliga, stora rapporten ”Svenskarna och internet” om svenskarnas användning av internet och dessemellan ett antal mindre studier.

Webbstjärnan
Webbstjärnan är en skoltävling som ger pedagoger och elever i den svenska grund- och gymnasieskolan möjlighet att publicera sitt skolarbete på webben. www.webbstjarnan.se

Internetmuseum
I december 2014 lanserade IIS Sveriges första digitala internetmuseum. Internetmuseums besökare får följa med på en resa genom den svenska internethistorien. www.internetmuseum.se

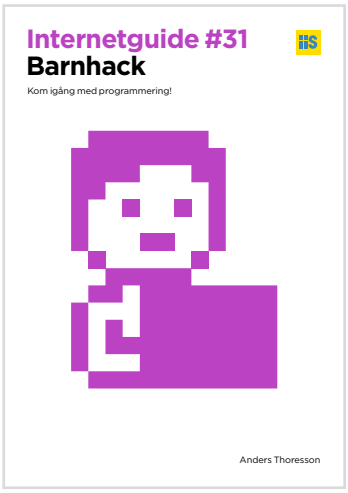
Federationer
En identitetsfederation är en lösning på konto- och lösenordshanteringen till exempel inom skolans värld eller i vården. IIS är federationsoperatör för Skolfederation för skolan och Sambi för vård och omsorg. www.iis.se/federation

Internets infrastruktur
IIS verkar på olika sätt för att internets infrastruktur ska vara säker, stabil och skalbar för att på bästa sätt gynna användarna, bland annat genom att driva på införandet av IPv6. www.iis.se

Sajtkollen
Sajtkollen är ett verktyg som enkelt låter dig testa prestandan på en webbsida. Resultatet sammanställs i en lättbegriplig rapport. www.sajtkollen.se

Läs mer på nätet redan idag! På Internetguidernas webbplats hittar du mängder av kostnadsfria publikationer. Du kan läsa dem direkt på webben eller ladda ner pdf-versioner. Det finns guider för dig som vill lära dig mer om webbpublicering, omvärldsbevakning, it-säkerhet, nätets infrastruktur, källkritik, användaravtal, barn och unga på internet, digitalt källskydd och mycket mer.

Nya Internetguider!



Barnhack
Kom igång med programmering!
Av: Anders Thoresson

Den här guiden är till för dig som vill genomföra egna programmeringskurser, eller laborera tillsammans med barnen hemma. Det är enklare än du tror och det krävs inte att du har teknik som ditt största intresse. Med vuxnas hjälp kan de unga bli aktiva producenter i stället för att vara passiva konsumenter. Du hittar mer material och videos på barnhack.se

Ungas integritet på nätet
Råd till dig som är vuxen
Av: Åsa Secher

Den här guiden belyser vad integritet på nätet betyder för unga idag. Det är ett viktigt ämne att ta sig an för att vi som vuxna ska kunna stötta och hjälpa barn och unga att känna var deras gränser går så att de inte råkar illa ut. Guiden riktar sig till vuxna i barns närhet och även om internetanvändandet börjar i tidig ålder, handlar innehållet framför allt om unga i åldrarna 10 till 16 år. Du får bland annat ta del av vad unga gör på nätet, vad grooming är, hur du som vuxen kan hjälpa unga att sätta gränser och vad Barnkonventionen säger om ungas rätt till integritet. Guiden är producerad i samarbete med Barnens Rätt i Samhället (BRIS).