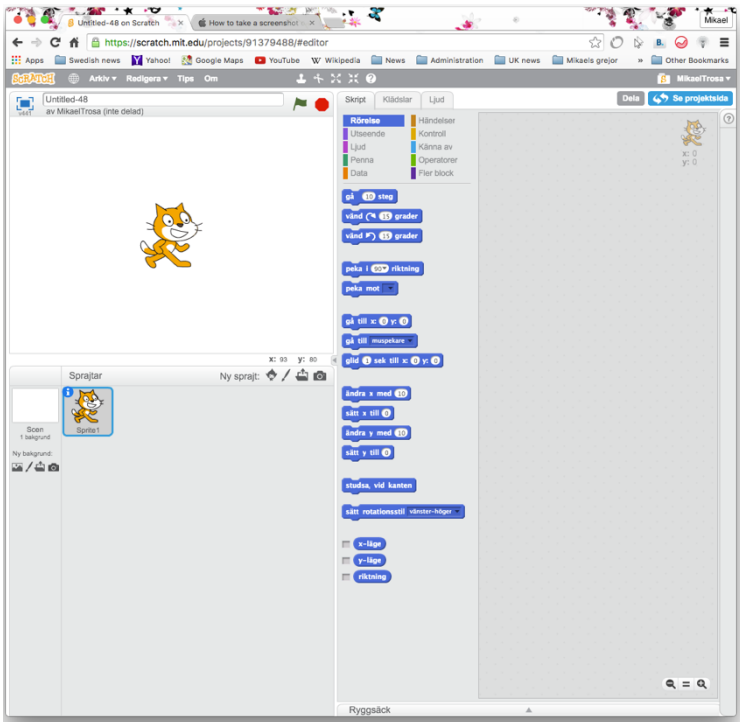
Välkommen till Scratch®!

Vi hoppas att denna publikation skall vara till glädje för barn och vuxna som tillsammans vill utforska Scratch och lära sig grunderna i programmering.

Gå in på https://scratch.mit.edu/ , välj språk och registrera dig som användare. Det är gratis och dina program sparas så att du kan komma åt dem från vilken internetansluten dator som helst.

Snabb rundtur

Nu skall du vara på Scratch hemsida. Om du ställt om till svenska, klicka på "Skapa". Om inte så ställ om till svenska genom att scrolla längst ner på sidan och välj svenska...



På denna sida finns följande:

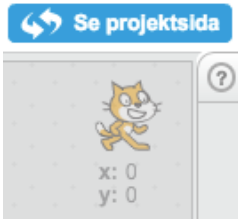
Scenen är den vita rutan där katten står. Här utspelar sig ditt spel eller din berättelse. I övre högra hörnet av scenen finns en grön flagga (klicka för att starta spelet eller berättelsen) och en röd cirkel (klicka för att stoppa allt). I övre vänstra hörnet av scenen finns en blå, fyrkantig symbol. Klicka på den så kommer scenen i fullskärmsläge.

Katten är den rollfigur som alltid kommer upp när du skapar ett nytt program. Rollfigurer i Scratch kallas sprite (eller på svenska - sprajt..) Du kan skapa flera rollfigurer och du behöver inte behålla katten. För att ta bort en sprajt så högerklickar (Mac ctrl-click) du på fyrkanten med sprajten och väljer radera. En ny sprajt skapas på fyra olika sätt - välj en från medföljande bibliotek, rita en egen, importera en sprite eller använd webbkameran för att använda en bild.

I mitten på skärmen hittar tre flikar - **Skript**, **Klädslar** och **Ljud**. Om du väljer fliken Skript så har du ett antal block som du använder för att göra dina program för den valda sprajten. Blocken drar du ut på den fria ytan längst till höger och där sätter du samman programmet för den valda sprajten. Flikarna **Klädslar** och **Ljud** gäller också bara för den valda sprajten och här kan du med olika klädslar skapa animerade figurer och importera eller skapa ljud för den valda sprajten.

Längst till höger är arbetsytan där du skapar dina program (under fliken **Skript**), dina klädslar, egna sprajtar eller bakgrunder (under fliken **Klädslar**) eller dina ljud (under fliken **Ljud**).

Hjälpmedel och verktyg



Längst upp till höger på skriptytan finns en rund knapp med ett frågetecken på. Klicka på den! Då får du tillgång till hjälp och tips under tre flikar, **Steg för steg**, **Hur gör man** och **Block**. Utforska de olika flikarna och de olika rubrikerna.

Nedan följer en översikt över de olika blocken i Scratch. För en detaljerad beskrivning, se http://wiki.scratch.mit.edu/wiki/Blocks.

Rörelseblocken hanterar förflyttning och rotation av sprajtar. Läge (x-läge, y-läge) och riktning är tillgängliga som variabler och används t ex om man vill beräkna avståndet till en annan sprajt.

Utseendeblocken hanterar byte av klädsel (t ex vid animering av rörelse), färg, storlek, gömma resp. visa sprajt. Man kan låta sprajten "prata" och "tänka". Vidare hanterar man i vilket bildlager som sprajten skall vara (framför eller bakom en annan sprajt?), storlek och färg på sprajten samt byte av bakgrund. Klädselnummer, bakgrundsnamn och storlek är tillgängliga som variabler.

Ljudblocken hanterar uppspelning av de olika ljud som importerats (klicka på Ljud fliken för att spela upp, spela in eller importera en ljudfil från ljudbiblioteket). Vidare kan man välja instrument (21 st!), justera volym och ändra tempo. Volym och tempo är tillgängliga som variabler.

Varje sprajt har en penna och denna penna hanteras via pennblocken. Man bestämmer färg, nyans och storlek. Man kan "stämpla" kopior av sprajten på scenen och rita spår som visar hur sprajten rört sig. Med raderablocket tar man bort allt som sprajten ritat på scenen.

De olika datablocken blir först tillgängliga när du skapar en variabel eller en lista. Variabler används för att hålla reda på olika saker, som poäng i ett spel eller om ett tillstånd är sant eller falskt. Variabler kan alltså vara både numeriska och alfanumeriska. Skapa en lista och ge den ett namn, t ex **godis**. Prova sedan att lägga in olika saker i **godis.** Hur kan du använda listor i dina spel eller berättelser?

Händelseblocken är de block som används för att starta de olika programmen. Startblocket med den gröna flaggan är kopplat till den gröna flaggan i övre högra hörnet på scenen. En användbar funktion är att använda meddelanden mellan programmen för att starta eller stoppa olika förlopp. Hur kan du använda detta?

Kontrollblocken används, dels för att repetera kommandon eller skript, samt för att testa olika villkor, som t ex om ett villkor är uppfyllt så skall skript A köras, om inte så körs skript B. Hanteringen av kloner (kopior av en sprajt) hittar du också bland kontrollblocken. Mer om kloner senare.

Blocken, som känner av olika händelser och tillstånd, kommer du att använda mycket. Om man vill styra en sprajt med tangenterna så använder du ett kontrollblock tillsammans med ett block som känner av när en viss tangent är nedtryckt. Om du vill att något skall hända när din sprajt hamnar på den gröna färgen på scenen så använder du blocket som känner av detta. Tillgängliga variabler är svar (på en fråga), ljudnivå (förutsätter mikrofon), x och y-läge på musen, video rörelse och riktning (förutsätter webbkamera), timer, x och y-läge av vald sprajt, aktuellt år, månad, datum, veckodag, timme, minut eller sekund, dagar sedan år 2000 och användarnamn. Du kanske vill ha olika scener i ditt spel beroende på veckodag?

Bland operatorerna finns både aritmetiska och logiska. Här kan du använda de fyra räknesätten, ha tillgång till matematiska funktioner (trigonometri, absolutvärde, min, max, kvadratrot, logaritmer och exponentialfunktioner. Du kan generera slumptal för att styra händelser i ditt spel. Du har logiska operatorer som "och", "eller" och "inte". Med hjälp av dessa operatorer kan du formulera villkor som styr händelserna i ditt spel eller din berättelse.

Under rubriken **Fler block** har man möjlighet att skapa egna block. Detta kan vara praktiskt om man använder samma skript på ett antal olika platser i koden. Man skapar då ett eget block som innehåller det skript som man vill återanvända.

Vidare finns det tillägg som låter Scratch hantera externa enheter,

- Picoboard (http://www.picocricket.com/picoboard.html)
- LEGO WeDo Construction Set (http://wiki.scratch.mit.edu/wiki/LEGO®_WeDo™_Construction_Set).

Scratch ritprogram (paint editor)

Ritprogrammet i Scratch har en mängd funktioner och vi kommer att visa en delmängd av dessa. Den kompletta beskrivningen hittar ni på http://wiki.scratch.mit.edu/wiki/Paint_Editor.

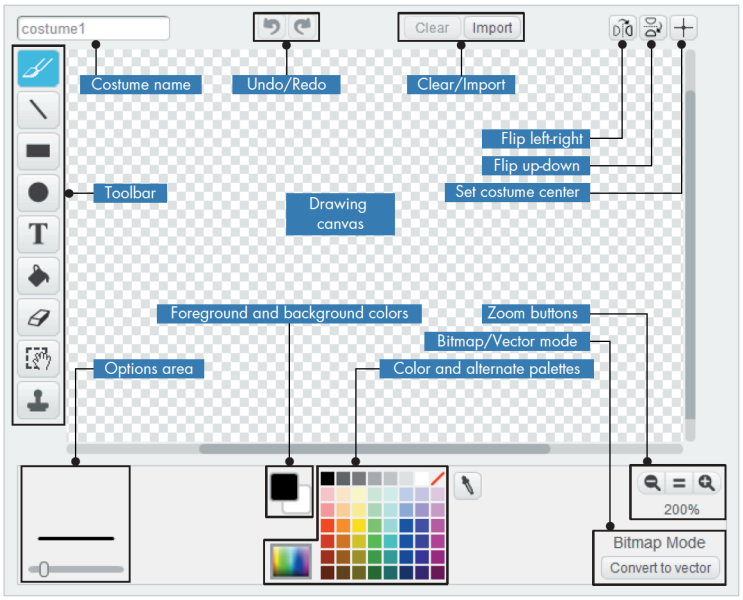
Det finns två huvudsakliga typer av bildfiler, bitmap och vektor. Bitmap formatet innebär att filen lagrar information om varje pixel, dess position och färg. Vanliga bitmap format är BMP, JPEG, PNG, GIF och TIFF. Vektor grafik innebär att man lagrar instruktioner eller regler för hur pixlarna skall ritas. Dessa regler använder punkter, kurvor och matematiska funktioner för att beskriva bilden. Vektorgrafik ser normalt mycket mera detaljerad ut när man zoomar in i bilden. Scratch använder SVG formatet (Scaleable Vector Graphics).

Ett sätt att beskriva skillnaden är att om du använder bitmap format och vill rita en cirkel på scenen så är det samma sak som att måla en cirkel med färg och pensel. När den är målad är den där och den går inte att flytta men den går att radera.

Om du använder vektorgrafik och vill rita en cirkel på scenen så skapar du en cirkel som du lägger på scenen. Nu kan du flytta cirkeln, ändra dess storlek eller deformera den. Din cirkel ligger alltså som ett objekt på scenen.

Vilka format man vill använda beror på vad man vill åstadkomma och vad man är van vid. Finessen är att Scratch hanterar båda sätten.

Scratch ritprogram i bitmap mode



Bitmap verktyg

Penseln är ett enkelt verktyg som målar med förgrundsfärgen där man klickar med musen. Pensels storlek justeras med regeln i **Options area.**

Linjeverktyget används för att dra raka linjer. Håll ner SHIFT tangenten för att dra perfekta vertikala eller horisontella linjer.

Rektangelverktyget används för att rita rektanglar (håll nere SHIFT tangenten för perfekt kvadrat). Man kan välja mellan ihålig eller fylld rektangel.

Ellipsverktyget används för att rita ellipser (håll nere SHIFT tangenten för perfekt cirkel). Man kan välja mellan ihålig eller fylld ellips.

Textverktyget används för att skriva text i en klädsel. Välj verktyget och klicka på ritytan och skriv in din text. Du kan ändra storlek och orientering men när du lämnar textverktyget så är texten en del av klädseln och kan inte flyttas.

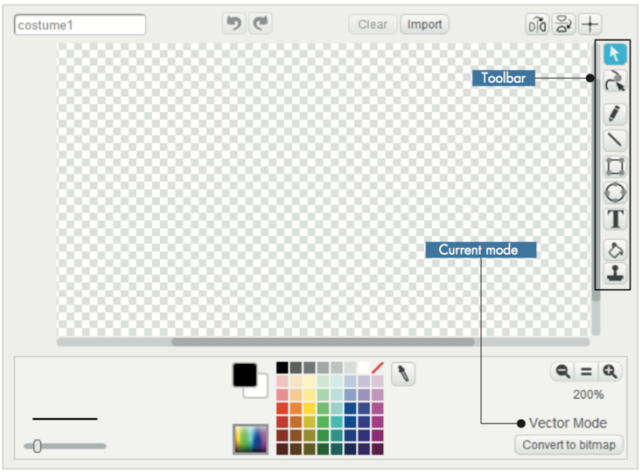
Fyll med färg används för att fylla en sluten region med samma färg, antingen en solid färg eller en gradient.

Raderaren tar bort/raderar det klickade området från ritytan. Färgerna ersätts med ingen färg/genomskinlig.

Väljverktyget används för att ta tag i ett område på ritytan och flytta, sträcka, rotera eller komprimera det.

Markera-och-kopiera används för att duplicera ett område på ritytan.

Scratch ritprogram i vector mode



Vector verktyg

Muspekaren används för att flytta, sträcka, komprimera eller rotera ett objekt. När ett objekt är valt markeras detta med en rektangel runt objektet.

Omformarverktyget har en mängd funktioner. Det kan böja och förändra formen på kurvor genom att ta tag i punkter på objektet och flytta dem. Läs mera om detta verktyg!

Ritverktyget i vector mode skapar kurvor som kan flyttas och böjas. Om man väljer en kurva med omformarverktyget kan man böja kurvan genom att ta tag i någon av punkterna på kurvan. Genom att klicka på kurvan får man en ny punkt som kan användas för att böja kurvan. Se även funktionen **släta ut** (nedre vänstra hörnet) som aktiveras när man valt en kurva eller objekt med omformarverktyget.

Linjeverktyget används för att dra raka linjer. Klicka och håll nere musen på startpunkten, dra till slutpunkten och släpp.

Rektangelverktyget används för att rita rektanglar. (håll nere SHIFT tangenten för perfekt kvadrat). Man kan välja mellan ihålig eller fylld rektangel.

Ellipsverktyget används för att rita ellipser (håll nere SHIFT tangenten för perfekt cirkel). Man kan välja mellan ihålig eller fylld ellips.

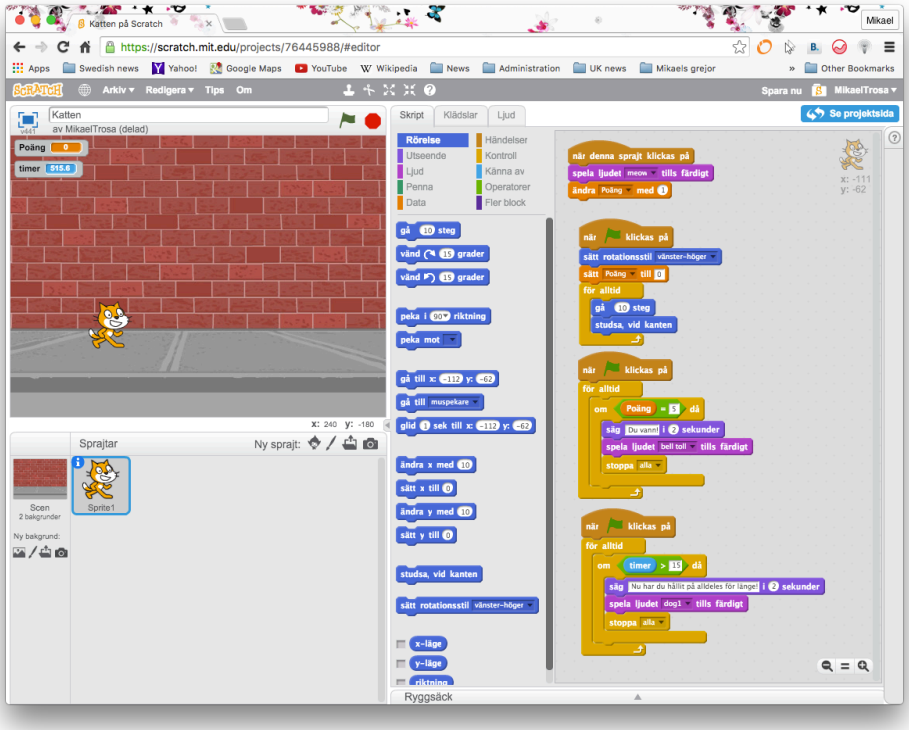
Textverktyget används för att skriva text i en klädsel. Välj verktyget och klicka på ritytan och skriv in din text. Till skillnad från textverktyget i bitmap mode så kan du editera texten fritt och textrutan förblir ett objekt på ritytan.

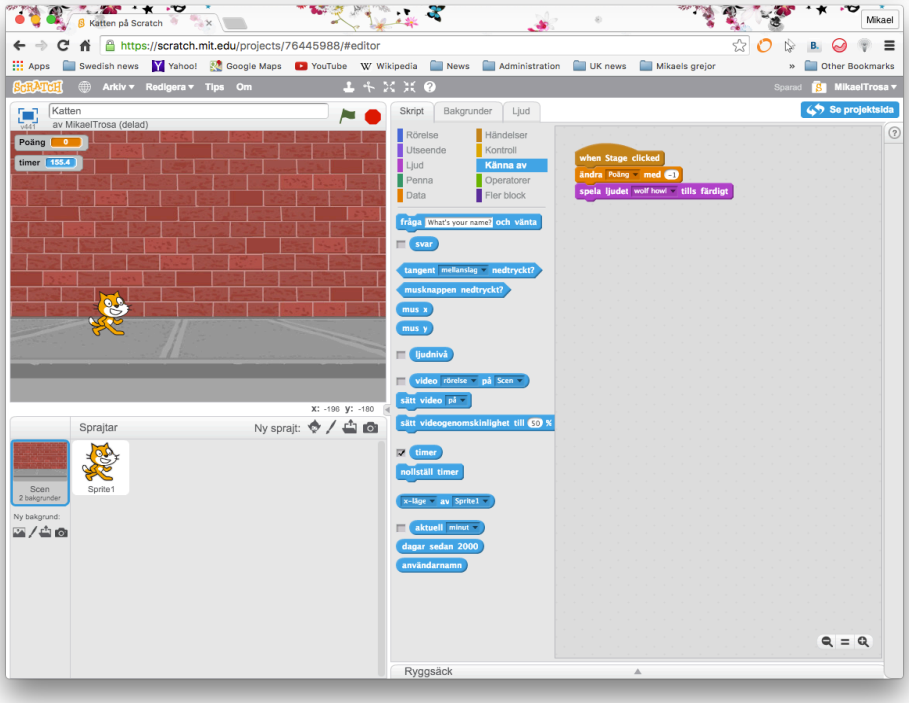
Fyll med färg används för att fylla ett objekt med samma färg, antingen en solid färg eller en gradient.

Markera-och-kopiera används för att kopiera ett objekt i vector mode.

Kattspelet

Det klassiska spelet där man skall försöka träffa katten med ett musklick. Träffar man får man 1 poäng, missar man får man minus 1 poäng. När man fått 5 poäng så har man vunnit och spelet stoppar. Om man inte fått 5 poäng inom 15 sekunder så har man hållit på för länge och förlorar.





Blocken i Kattspelet

Blocket som får katten att röra sig

Vi är nu på den skriptyta som tillhör katten!

Som god programmerare ser man till att sätta alla variabler och alternativ rätt innan programmet startat.

Startblocket med den gröna flaggan kommer först.

Katten har 3 möjliga rotationsstilar och det är vänster-höger som vi vill ha. Pröva att ändra rotationsstil för att se vad som händer! Var annars kan man ställa rotationsstil?

Skapa variabeln **Poäng**! Vi ser till att poängen sätts till noll innan spelet startas.

För att få katten att röra sig använder vi en **för alltid** slinga (loop) där vi säger till katten att gå 10 steg, och om katten då slagit i kanten skall den studsa, dvs byta riktning. Sen gör den det igen och igen precis som vi vill. Pröva att öka eller minska antalet steg och se vad som händer!

Blocket som får katten att jama när vi klickar på den

Här använder vi ett annat block som känner att när vi klickar på katten (denna sprajt). Vi ser till att spela ljudet meow och när det är färdigt ändrar vi poäng med +1. Vi har tidigare skapat variabeln poäng med hjälp av det gula data blocket.

Blocket som kontinuerligt kontrollerar

Detta block övervakar hur många poäng man för tillfället har och, om poängen blir 5, säger "**Du Vann!**", spelar ljudet **bell toll** och stoppar spelet.

Här har vi vår första **slinga med ett villkor** tillsammans med en **för alltid slinga.**

Den här konstruktionen kommer du att använda jämt.

I den innersta slingan har vi ett villkor som säger att om poäng är lika med 5 så skall de följande instruktionerna genomföras. Med hjälp av den yttre **för alltid slingan** ligger vi och frågar kontinuerligt ända till villkoret är uppfyllt. Hur skulle programmet fungera om vi tog bort **för alltid slingan**?

Blocket som säger ifrån att vi har hållit på för länge

Här har vi också ett block med ett villkor och ett **för alltid block** utanpå. I Scratch finns det en timer som går kontinuerligt men som nollställs när man klickar på den gröna flaggan. Här har vi nu satt villkoret att om timer har gått **mer än (>)** 15 sekunder så skall de följande instruktionerna genomföras. Utanpå den slingan sitter då återigen för alltid slingan och kör skriptet gång på gång till villkoret är uppfyllt. Varför har vi valt **mer än (>)** istället för **lika med(=)**? Prova och byt och se vad som händer!

Vi går nu över till skriptytan för scenen

Blocket som säger ifrån om vi missar katten

Om vi missar katten så träffar vi scenen. Det här blocket gör att vi får minuspoäng när vi missar katten (vilket är samma sak som att träffa scenen). Dessutom spelar vi ett lämplig ljud. Som bakgrund har vi valt en bakgrund från biblioteket, **brick wall 1.**

Sammanfattning

Det färdiga spelet består av totalt 5 skript eller program. Fyra av dessa finns på kattens skriptyta och ett på scenens skriptyta. Vilka av dessa skript fungerar bara på resp skriptyta och vilka kan vara på vilken skriptyta som helst?

I det här spelet har vi lärt oss:

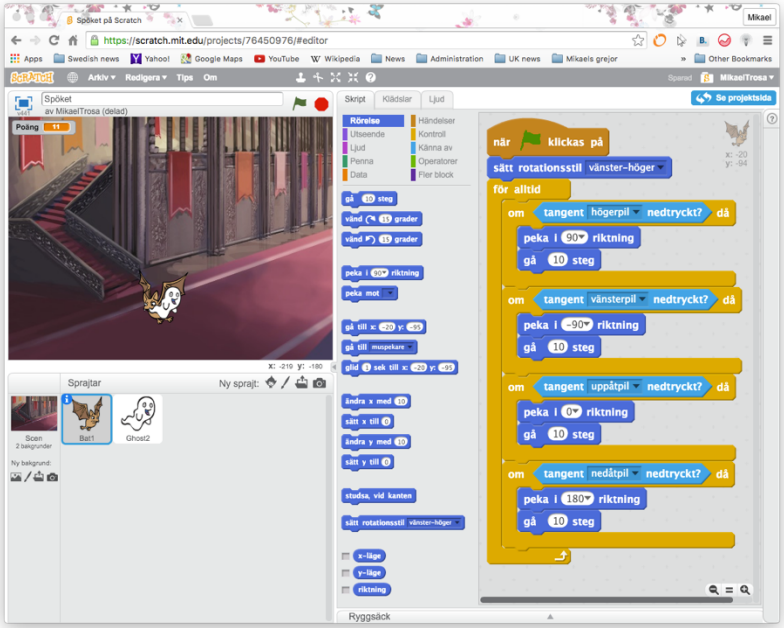
- Hur vi får katten att springa fram och tillbaka med hjälp av ett **rörelseblock** och en **för alltid slinga**
- Hur vi skapar en **variabel** som blir våra poäng och hur vi nollställer variabeln när vi startar spelet och som ökar med 1 när vi träffar katten och minskar med 1 när vi träffar scenen (=missar katten)
- Hur vi använder **villkor** kombinerat med **för alltid slinga** för att övervaka, dels poängställningen (så att vi kan säga att vi vunnit, dvs fått 5 poäng innan tiden (15 sek) gått ut), dels att spelet stoppas efter 15 sekunder.

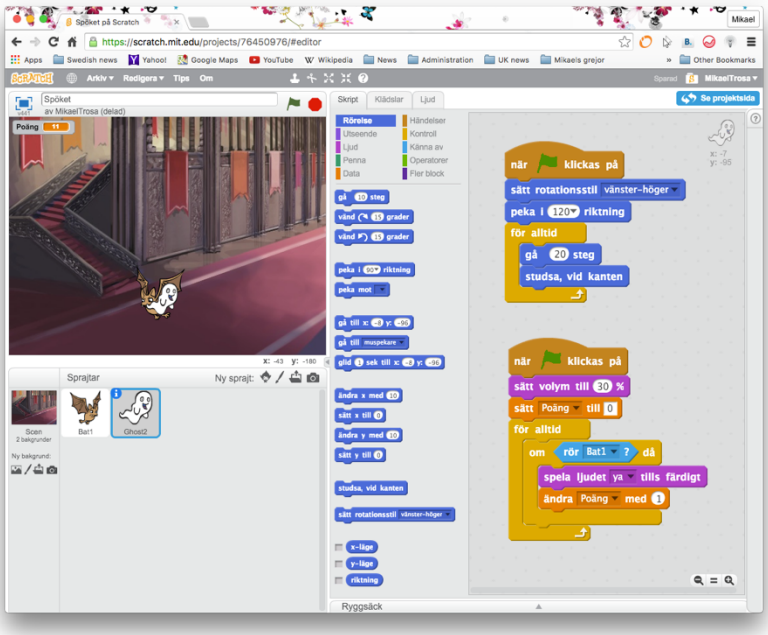
Spelet laddas ned från http://bredbandarna.se/spel/Kattspelet.sb2. Gå till Arkiv på Scratchmenyn och välj "Ladda upp från din dator" och välj filen du just laddat ned. Du kan också använda följande länk: https://scratch.mit.edu/projects/92553922/

Spöket

I det här spelet åker ett spöke runt och vi styr fladdermusen med piltangenterna. När spöket och fladdermusen träffar varandra spelas ett ljud och man får en poäng.

Det som vi skall lära oss i detta spel är hur man kan styra sprajtar med tangenter och hur man känner av när två sprajtar kolliderar. Var och en kan sedan bygga vidare på spelet med andra funktioner och utmaningar.





Blocken i Spöket

Blocket som får fladdermusen att röra sig

Vi är nu på den skriptyta som tillhör fladdermusen

Först ställer vi rätt rotationsstil för fladdermusen.

Vi använder de ljusblå **känna av** blocken, en **slinga med villkor** för varje tangent och sedan en **för alltid slinga** utanpå alltihopa för att kontinuerligt kontrollera om någon tangent är nedtryckt.

Pröva vad som händer om du trycker på flera tangenter samtidigt!

Blocket som hanterar spöket

Vi är nu på den skriptyta som tillhör spöket

Återigen ställer vi rotationsstil. Till skillnad från kattspelet där katten bara gick fram och tillbaka vill vi att spöket skall röra sig över större delen av scenen och därför bestämmer vi att när spöket startar skall det peka i riktning 120 grader. Prova att ändra detta och se vad som händer!

Därefter samma lösning som i kattspelet, gå ett antal steg, om du rör kanten, studsa (byt riktning) och sedan en **för alltid slinga** utanpå alltihopa.

Bakgrund från biblioteket, **castle 4**

Blocket som hanterar att känna av kollisioner och räkna poäng

Vi är fortfarande på skriptytan som tillhör spöket

Först sätter vi ljudvolymen till 30% (det valda ljudet **ya** var ganska påträngande..) och sedan nollställer vi variabeln **Poäng**.

Vi har sedan den vanliga lösningen med en **slinga med villkor** och utanpå den en **för alltid slinga**.

Villkoret är att, om fladdermusen (Bat1) rör den *sprajt som skriptytan tillhör*, så skall instruktionerna i slingan med villkor utföras.

För alltid slingan utanpå detta ser till att kontinuerligt fråga om villkoret är uppfyllt och när det sker spelar ljudet och poäng ökas med 1.

Pröva att byta ut blocket **spela ljudet ya tills färdigt** mot blocket **spela ljudet ya** och se vad som händer med poängräkningen. Du får flera poäng för varje träff! Varför??

I det här spelet har vi lärt oss:

- Hur vi kan styra rörelser i spelet med hjälp av tangenter. Om man vill att en spelare skall sköta fladdermusen och en annan spelare skall sköta spöket så kan man på precis samma sätt göra så att t ex tangenterna w, a, s och z styr spöket och så kan man jaga varandra. Pröva!
- Hur vi kan känna av när 2 sprajtar rör vid varandra (kolliderar?). Utforska **känna av** blocken!

Spelet laddas ned från http://bredbandarna.se/spel/Spoeket.sb2. Gå till Arkiv på Scratchmenyn och välj "Ladda upp från din dator" och välj filen du just laddat ned. Du kan också använda följande länk: https://scratch.mit.edu/projects/92554002/

Avstånd & riktning

Inledning

Denna kodmodul beräknar kontinuerligt avståndet mellan 2 sprajtar samt låter den ena sprajten peka mot den andra. Vi kommer att använda denna kodmodul i ett Skattjaktspel.

Blocket som beräknar avståndet mellan sprajtarna

Vi är nu på den skriptyta som tillhör scenen.

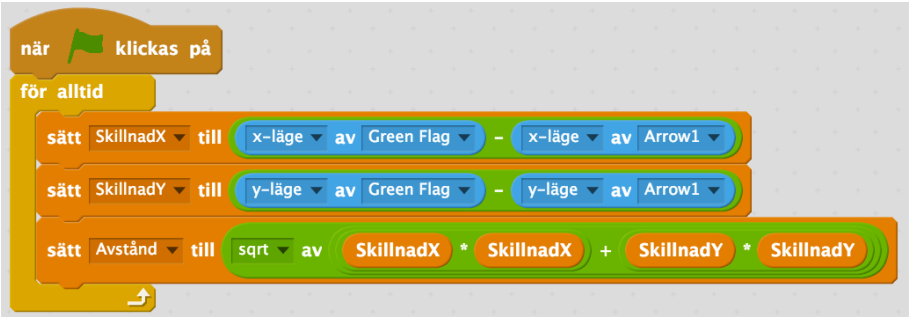
Avståndet beräknas genom Pythagoras sats, se:

http://www.matteboken.se/lektioner/skolar-9/geometri/pythagoras-sats

Vi skapar först tre variabler, **Avstånd**, **SkillnadX** och **SkillnadY**.

Dessa tre variabler utgör sidorna i en rätvinklig triangel. Vi beräknar SkillnadX resp SkillnadY som skillnaden mellan sprajtarnas **x-lägen** och **y-lägen**. Därefter beräknar vi avståndet enligt Pythagoras sats som:

Kvadratroten ur summan av SkillnadX gånger SkillnadX plus SkillnadY gånger SkillnadY.



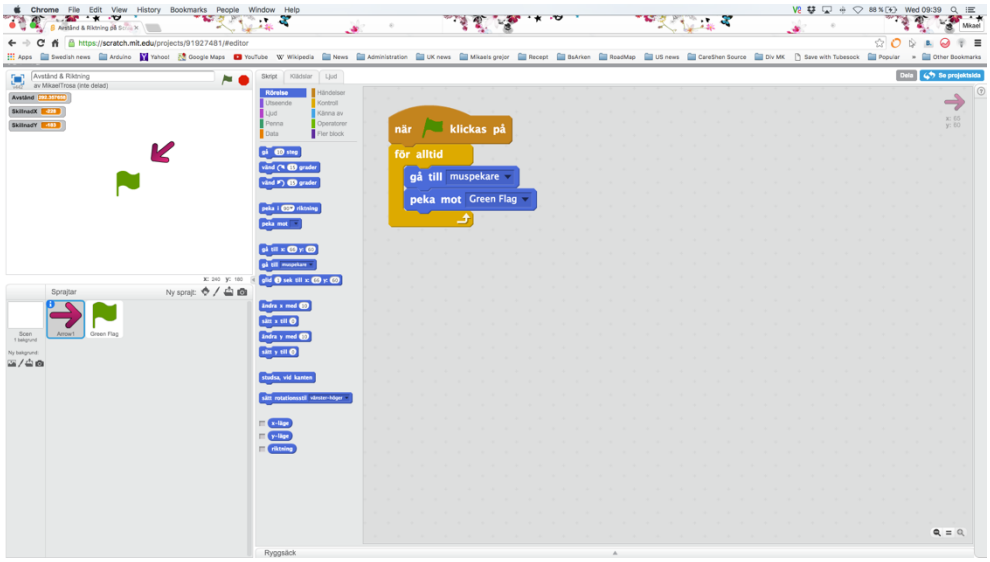
Genom **för alltid slingan** beräknas avståndet med sprajtarna kontinuerligt.

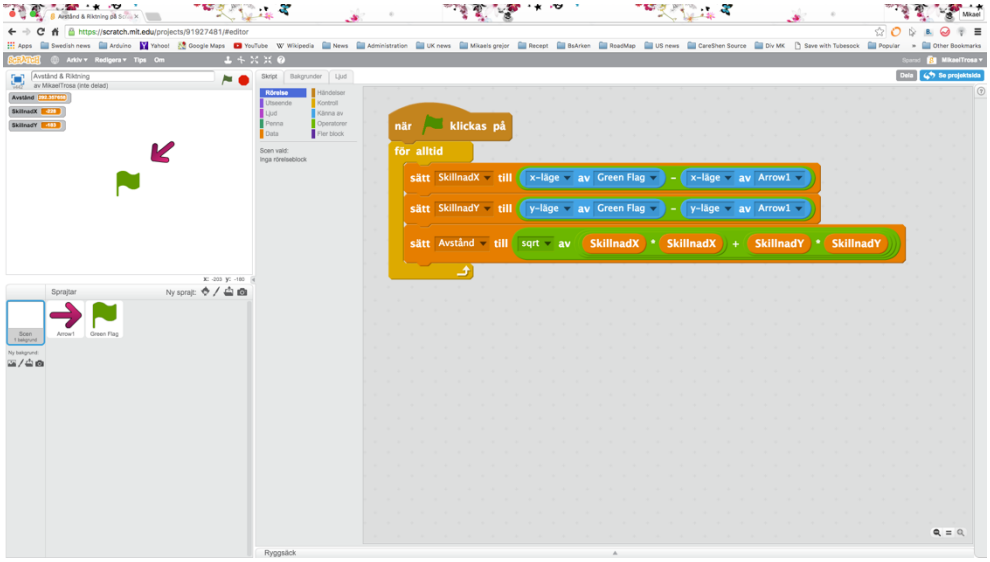
Blocket som styr sprajten Arrow1 och ser till att sprajten Arrow1 alltid pekar på sprajten Green Flag

Vi är nu på den skriptyta som tillhör Arrow 1.

Genom blocket "**gå till muspekare**" så kommer sprajten att gå till muspekaren. Blocket "peka mot Green Flag" ser till att sprajten riktas mot Green Flag. Med hjälp av **för alltid slinga** kommer detta att ske kontinuerligt.

Den kompletta koden ser ut så här:





I det här spelet har vi lärt oss:

- Hur man bygger formler med hjälp av **variabler** och **operatorer**
- Hur man med en **för alltid slinga**, kontinuerligt beräknar en formel för att visa avståndet mellan två sprajtar

Spelet laddas ner från http://bredbandarna.se/spel/Avstaand och Riktning.sb2. Gå till Arkiv på Scratchmenyn och välj "Ladda upp från din dator" och välj filen du just laddat ned. Du kan också använda följande länk: https://scratch.mit.edu/projects/92554027/

Skattjaktspelet

Här letar vi dolda skatter, kanske på planeten Mars. Med hjälp av vår skattletarsprajt så får vi reda på riktningen till skatten och avståndet märker vi genom att ljudet blir starkare ju närmare skatten vi kommer. När vi är tillräckligt nära anser vi att skatten är hittad, vi får ett poäng och en ny dold skatt placeras ut som vi skall hitta.

Blocken i Skattjaktspelet

Vi skall använda vår Avstånd & Riktning modul som bas. Börja med att ladda ned programmet från:

http://bredbandarna.se/spel/Avstaand och Riktning.sb2. Gå sedan till Arkiv på Scratchmenyn och välj "Ladda upp från din dator". Döp sedan om spelet till Skattjaktspelet och spara. Du kan också ladda ner modulen genom denna länk: https://scratch.mit.edu/projects/92554027/

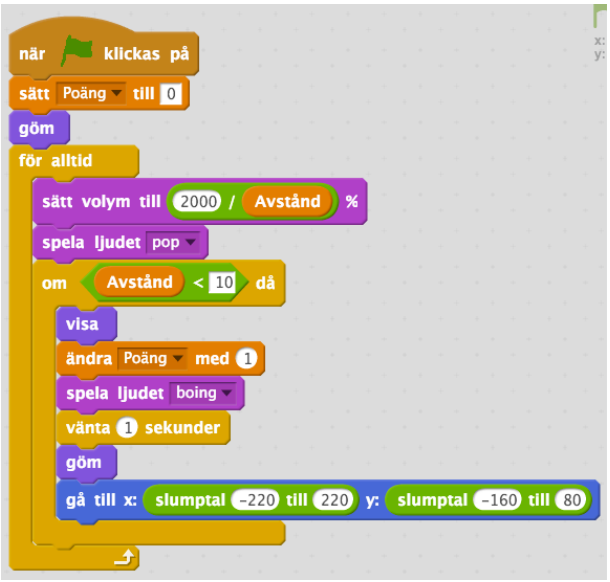
Ladda ned ny bakgrund till scenen, t ex **moon** som används i vårt exempel.

Vi har då koden för såväl Scenen som sprajten Arrow1 klar. Nu skall vi skapa och gömma skatten.

Blocket som skapar en skatt, gömmer den och placerar ut den slumpmässigt

Vi är nu på den skriptyta som tillhör Green Flag!

Blocket hanterar poäng, justerar volymen på ljudet, övervakar avståndet till skatten och upptäcker när vi är mindre än 10 enheter från skatten och då visar skatten, ger 1 poäng, spelar ett ljud som visar att vi hittat skatten, gömmer skatten och placerar ut en ny skatt på en slumpmässigt vald plats.



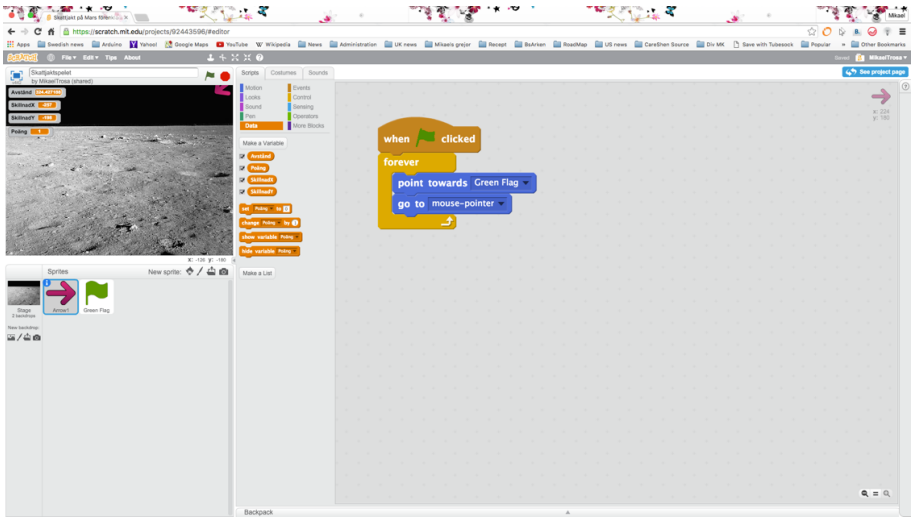
Vi börjar med att sätta poäng till nol och gömma Green Flag. Det första blocket i **för alltid** slingan justerar volymen på ljudet pop så att volymen är omvänt proportionell till avståndet. Värdet av uttrycket (2000 delat med avståndet) blir ju större ju mindre avståndet är.

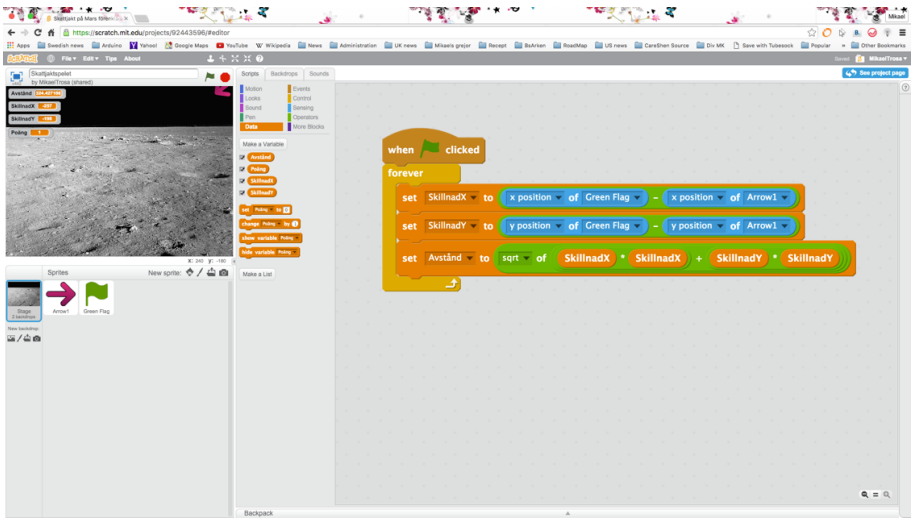
Vi har sedan **slingan med villkor** och villkoret säger att om avståndet till skatten är mindre än 10 enheter så skall instruktionerna i slingan utföras.

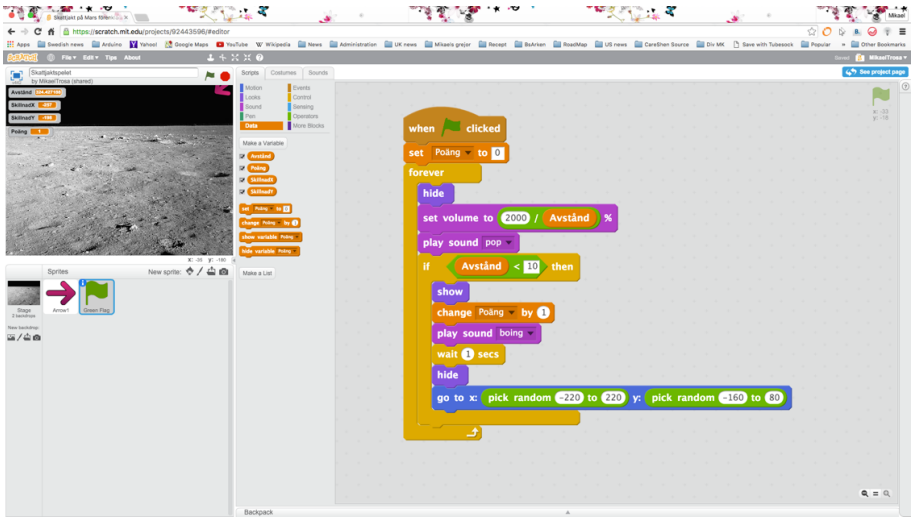
När detta händer så visas Green Flag, ett poäng läggs till, ljudet **boing** spelas, en paus på en sekund

Sedan göms Green Flag igen och placeras ut slumpmässigt inom ett område där x-koordinaten är mellan -220 och +220 och y-koordinaten är mellan -160 och +80 (vilket är ungefär månytan) och jakten på nästa skatt kan börja.

Den kompletta koden ser ut så här:







I det här spelet har vi lärt oss:

- Hur man åstadkommer att ljudet ökar när avståndet minskar (omvänt proportionell)
- Ytterligare en variant på kombinationen av **slinga med villkor** (avstånd mindre än 10) och en **för alltid slinga** som kontinuerligt frågar **slingan med villkor**
- Hur man använder slumptalsfunktionen för att placera ut skatten

Spelet laddas ner från http://bredbandarna.se/spel/Skattjaktspelet.sb2. Gå till Arkiv på Scratchmenyn och välj "Ladda upp från din dator" och välj filen du just laddat ned. Du kan också använda följande länk: https://scratch.mit.edu/projects/92554051/

Dubbelhopp

Denna kodmodul visar hur man kan göra en trovärdig animering av en katt som hoppar. Dessutom innehåller koden en funktion som begränsar möjligheten till ett antal upprepade hoppförsök. Katten måste landa efter ”dubbelhopp” innan den kan hoppa igen.

Kod

Vi bygger koden steg för steg.

Kattens rörelse höger och vänster

Kontrolleras med piltangenterna. En enkel animering med byte av klädsel ger intryck av att katten springer. Kattens rotationsstil sätts i koden.

Kattens begränsningar och animering av fall nedåt

Katten skall inte kunna komma lägre ned än **y=minus 130**. Detta sker i **Om** villkoret i kodblocket som säger att om:

och då skall dessa rader köras

Vi vill att katten skall falla med ökande hastighet när den befinner sig högre än **y=minus 129**.

Detta hanteras genom att vi skapar variabeln **fallhastighet** som vi använder för att bestämma hur mycket kattens **y-läge** skall ändras varje gång som programmet går igenom slingan "**annars, ändra fallhastighet, ändra y**".

Detta kodblock fungerar som följer:

Släpp katten varsomhelst på scenen (med musen).

Om-villkoret **y<minus 129** är då inte uppfyllt, utan det är **Annars**-villkoret som gäller.

Variabeln **fallhastighet** ändras med minus 2 (utgångsläget var 0) och blir minus 2. Kattens **y-läge** ändras därför med minus 2.

Nästa gång som slingan körs ändras **fallhastighet** återigen med minus 2 och blir minus 4.

Kattens **y-läge** ändras nu med minus 4.

På detta sätt kommer katten **y-läge** att ändras med större och större steg tills slutligen **Om**-villkoret **y<minus 129** uppfylls och katten då placeras på **y=minus130**

Katten hoppar på kommando

Vi vill nu åstadkomma en trovärdig animering av ett hopp där katten hoppar upp och hastigheten uppåt avtar för att sedan bli noll och sedan börja falla nedåt med ökande fart.

Vi har redan löst problemet med att falla med ökande fart så nu skall vi bara lösa problemet med att få katten att hoppa upp.

Detta gör vi med ett kodblock som fungerar som följer:

När vi trycker på mellanslag så sätter vi variabeln **fallhastighet** till 20. Observera att detta steg inte påverkar katten. Det är i nästa rad där vi **ändrar y med 20** som rörelsen uppåt börjar.

Nu har detta kodblock gjort sitt och nu det är vårt redan färdiga kodblock med slingan "**annars, ändra fallhastighet, ändra y**" som tar kontroll (eftersom villkoret att **y<minus 129** inte längre är uppfyllt!!).

När den slingan börjar arbeta är **fallhastighet** lika med 20 och kattens **y-läge har ökat med 20**.

Nu minskas **fallhastighet** med 2, dvs nu är den 18, **y** ökar med 18, dvs katten fortsätter uppåt. Nästa gång minskar **fallhastigheten** igen med 2 och blir 16, **y** ökar med 16 och katten fortsätter uppåt med 16.

Detta fortsätter och **fallhastighet** kommer så småningom att bli noll och katten kommer att börja falla till **y< minus 129** och sedan stanna på **y= minus 130** och **fallhastighet** sätts till 0.

Nu hoppar katten på kommando men vi vill nu begränsa så att den bara kan göra två hopp i följd utan att landa på "marken".

Det gör vi nu i nästa avsnitt.

Dubbelhopp

Katten kan nu hoppa hur många gånger som helst utan att landa på fötterna och det skall vi sätta stopp för.

För att åstadkomma detta behöver vi hålla reda på om katten har hoppat ett andra hopp utan att landa på fötterna och det gör vi med hjälp av en variabel som vi kallar **dubbelhoppat** och den har två värden, **ja** och **nej**.

Vi börjar med att göra en första ändring i koden som följer:

Från:

Till:

Prova och se vad som hänt! Nu går det bara att göra enkelhopp! Vi har satt ett villkor att katten måste stå på backen (**y= minus 130**) för att två blocken i slingan skall köras.

Tillåta ett andra hopp men sedan ner på backen

Vi skall nu lägga till kod i **annars** slingan som tillåter ett hopp där katten inte står på backen under förutsättning att katten bara gjort ett hopp sedan han stod på backen.

Vi utgår från koden ovan och lägger till ytterligare en **om** slinga:

Om katten står på backen (y= minus 130) så körs programmet som vanligt.

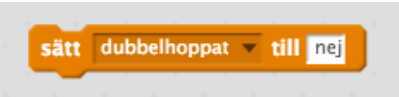
Om katten däremot inte står på backen (vilket är fallet om katten just hoppat och man trycker på mellanslag igen innan katten landat) så kommer **annars** slingan att köras.

Om då variabeln **dubbelhoppat = nej** kommer slingan att köras som tidigare men med skillnaden att variabeln **dubbelhoppat sätts till ja**.

Programmet går vidare som tidigare med **annars** slingan som ändrar **fallhastighet** med minus 2 och sedan ändrar y med **fallhastighet.**

Men en viktig sak skiljer - nu har variabeln **dubbelhoppat** ställts om till **ja** vilket betyder att om tangenten **mellanslag** trycks ner och katten fortfarande är i luften så kommer detta inte att leda till något alls (**y-läget är inte minus 130** och **dubbelhoppat är lika med ja**).

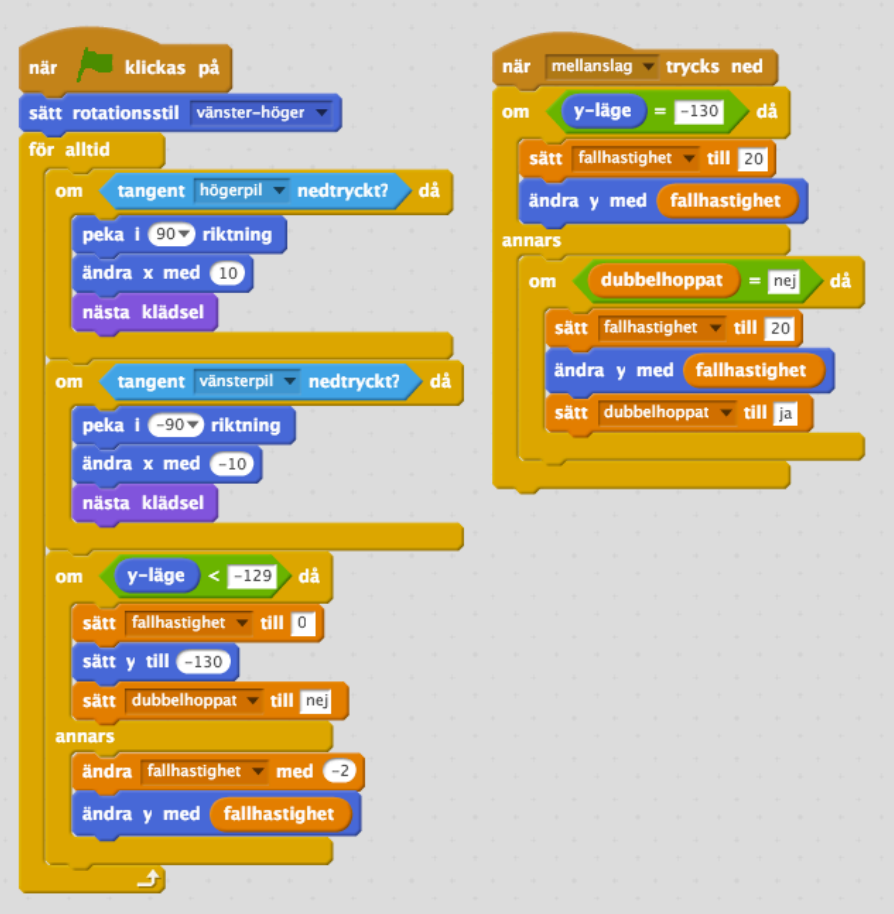
Testa nu att hoppa men sätt först variabeln **dubbelhoppat till nej**.



Du kommer att kunna göra ett dubbelhopp och sedan bara enkelhopp tills du än en gång **satt dubbelhoppat till nej**.

Sista kodraden

Genom att i **om slingan**, som känner av när **y är mindre än minus 129**, lägga till sätt **dubbelhoppat till nej** så gör vi det möjligt att dubbelhoppa igen efter det att katten landat på backen och vårt program är komplett.



Spelet laddas ner från http://bredbandarna.se/spel/Dubbelhopp.sb2. Gå till Arkiv på Scratchmenyn och välj "Ladda upp från din dator" och välj filen du just laddat ned. Du kan också använda följande länk: https://scratch.mit.edu/projects/92554067/

Originalet finns på:

https://scratch.mit.edu/projects/11708677/

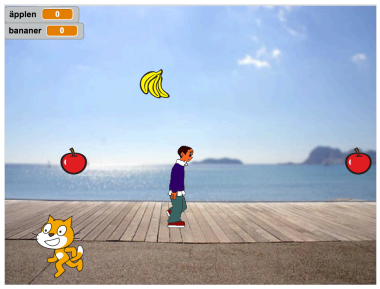
Youtube lektion finns på:

https://www.youtube.com/watch?v=oRBfjK-qeXE

Katten hoppar

Här nyttjar vi dubbelhopp modulen för att låta katten hoppa och plocka äpplen och bananer.

I detta spel kommer vi att använda kloner, slumptal och koordinater.



Kod

Vi bygger koden steg för steg.

Vi börjar med att använda dubbelhopp modulen. Du kan ladda ner dubbelhoppmodulen från:

http://bredbandarna.se/spel/Dubbelhopp.sb2

Du kan också använda länken:

https://scratch.mit.edu/projects/92554067/

Kloner och koordinater

Spelet går ut på att katten skall springa fram och tillbaka och hoppa upp och plocka äpplen och bananer. Katten styrs i sidled av piltangenterna och hoppar med mellanslagstangenten, precis som i dubbelhopp modulen. Börja med att ladda ner scenbakgrunden **boardwalk.**

Äpplen

Hämta en ny sprite (Apple). Vi skall göra kloner (kopior) av denna sprite och låta dessa kloner uppträda slumpmässigt och sedan försvinna. Katten måste hinna ta dem innan de försvinner.

När vi vill använda kloner så måste man hantera två olika saker. Man måste bestämma **när de skall uppstå** och hur de skall **bära sig åt under den tid som de existerar**.

Ett praktiskt råd är att ha koden, som bestämmer när klonerna skall uppstå, på skriptområdet för scenen.

Koden som bestämmer hur klonerna skall uppträda lägger man på skriptområdet för spriten som man klonar.

Skript som styr hur äpplet beter sig sedan det skapats

Det första som sker är att man gömmer spriten för den kommer inte att delta, endast kloner (kopior) av spriten kommer att delta.

Dessa block ligger på skriptområdet för äpplet!

Nästa steg är att bestämma vad som skall ske när man startar klonen. Här vill vi placera äpplena på den höjd som katten når med ett enkelhopp och det motsvarar **y = minus 20**.

Vi vill sedan slumpmässigt placera äpplet i sidled och det gör vi genom att använda slumptalsfunktionen och skapa ett tal mellan minus 220 och plus 220. Äpplet kommer då slumpmässigt att skapas någonstans mellan ytterkanterna.

Äpplet skall sedan försvinna och här använder vi också slumptalsfunktionen för att skapa en väntetid (livslängd) på mellan 2 och 5 sekunder. Sedan gömmer vi klonen och raderar den.

Nästa kodblock kontrollerar vad som händer när vår sprite och äpplet rör vid varandra. Då ändrar vi variabeln **äpplen** med 1, spelar ett ljud och raderar klonen.

Skript som bestämmer när klonen skall skapas

Detta skript ligger på scenens skriptområde!

På detta sätt skapar man en klon med slumpmässigt intervall mellan 1 och 5 sekunder. Vi ser alltså att man använder minst 2 olika program för att hantera kloning; ett som bestämmer uppträdandeet och ett som bestämmer när/att klonen skall skapas.

Bananer

Dessa hanteras på exakt samma sätt. Skillnaden är att bananerna placeras högre så att katten måste dubbelhoppa för att nå dem (**sätt y till 80**). Vi har också importerat ett annat ljud.

Dessa block ligger på skriptområdet för bananen!

Skriptet som startar klonen har samma struktur och **ligger på scenens skriptområde**.

Gående figur

För att illustrera animering så sätter vi in en figur som redan har ett antal klädslar och som därför lämpar sig för animering. Importera ny sprite och välj "Jaime Walking".

Koden på dennes skriptområde ser ut som följer:

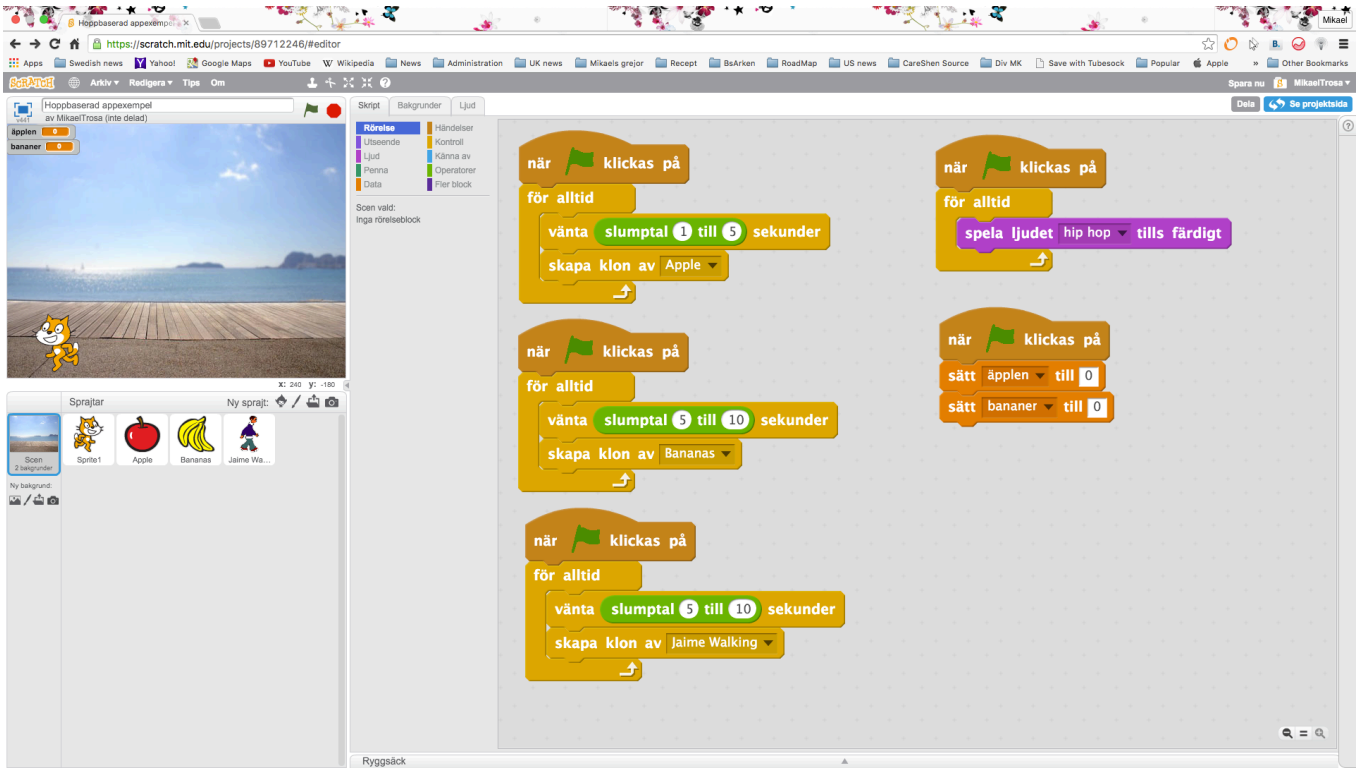
Koden på scenens skriptområde är gjord på samma sätt som koden för äpplet och bananen.

Bakgrundsmusik & nollställning

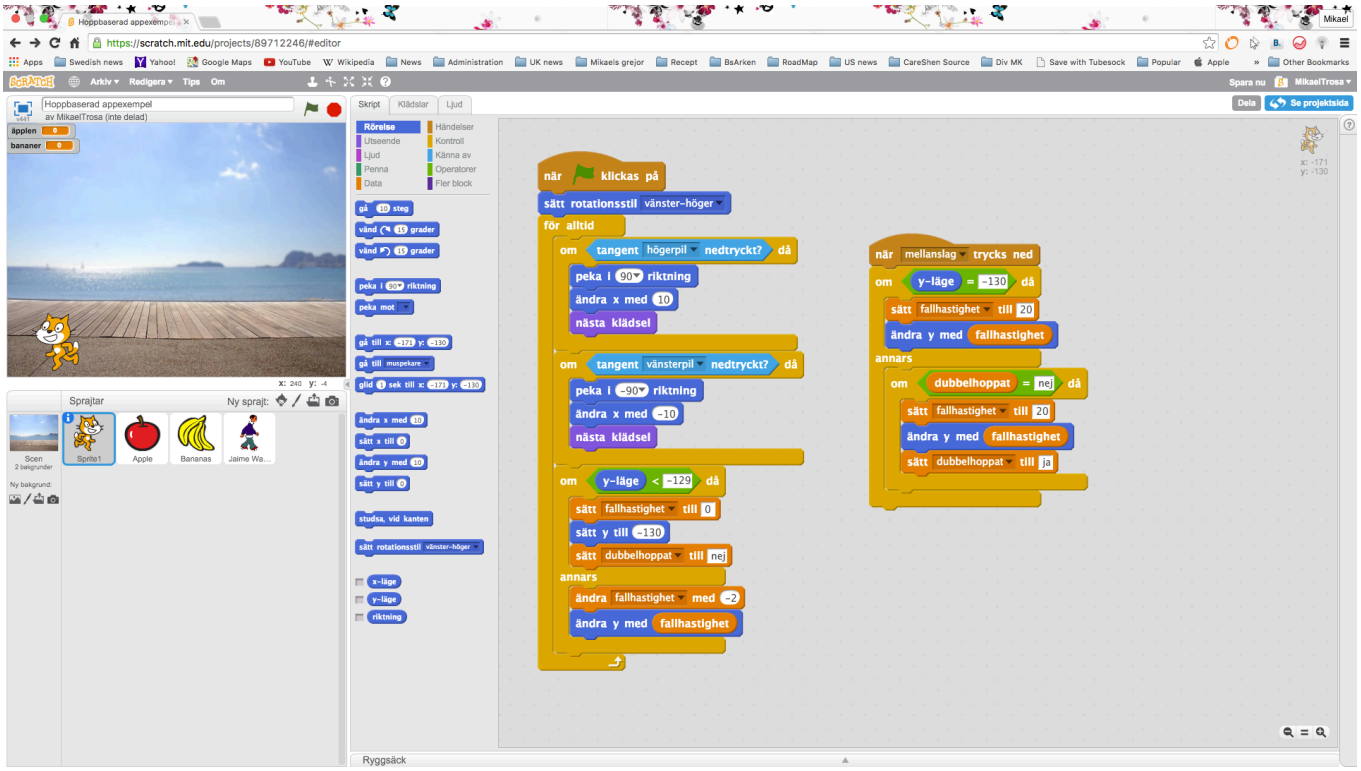
2 ytterligare kodblock på scenens skriptområde och sedan är spelet färdigt - ett program för bakgrundsmusik, och ett för att nollställa poäng när spelet startar. Ladda ned ljudet hip hop!

Det kompletta spelet

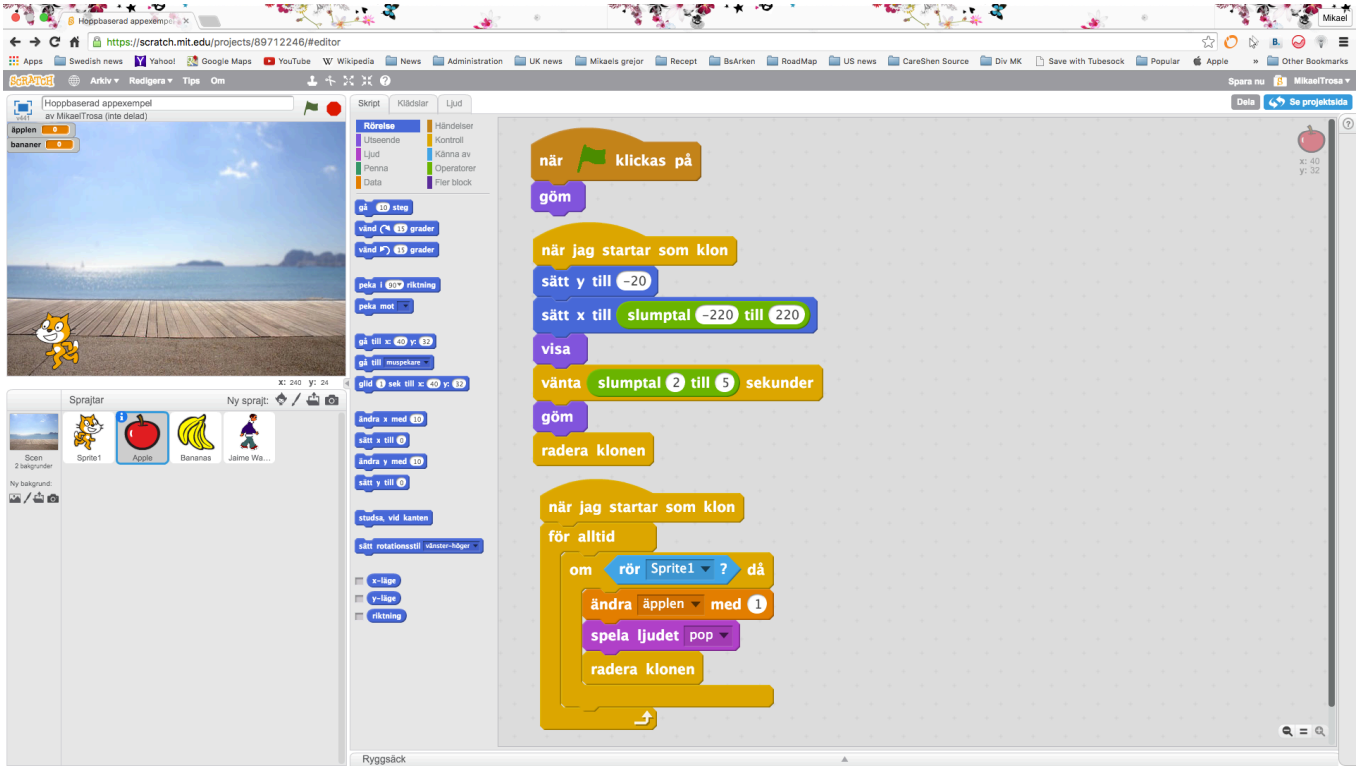
Scenen



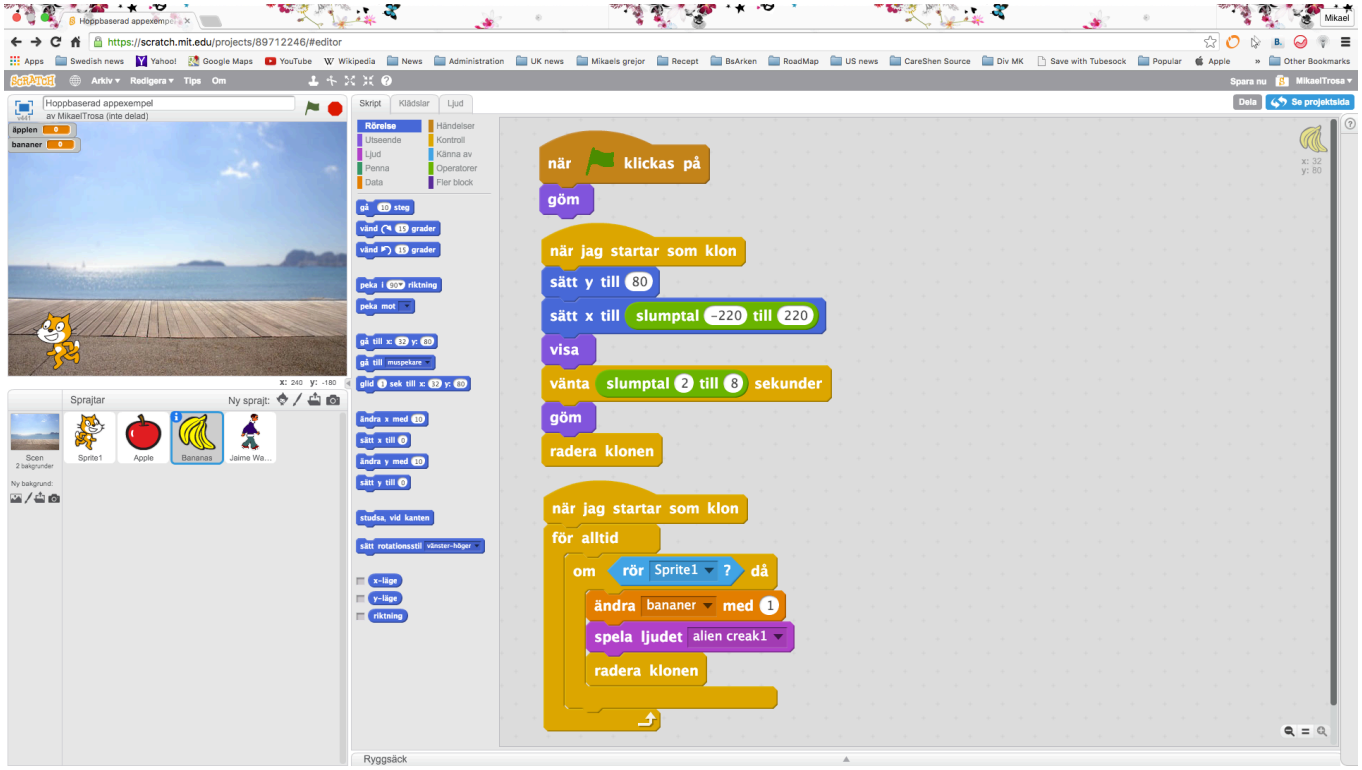
Katten



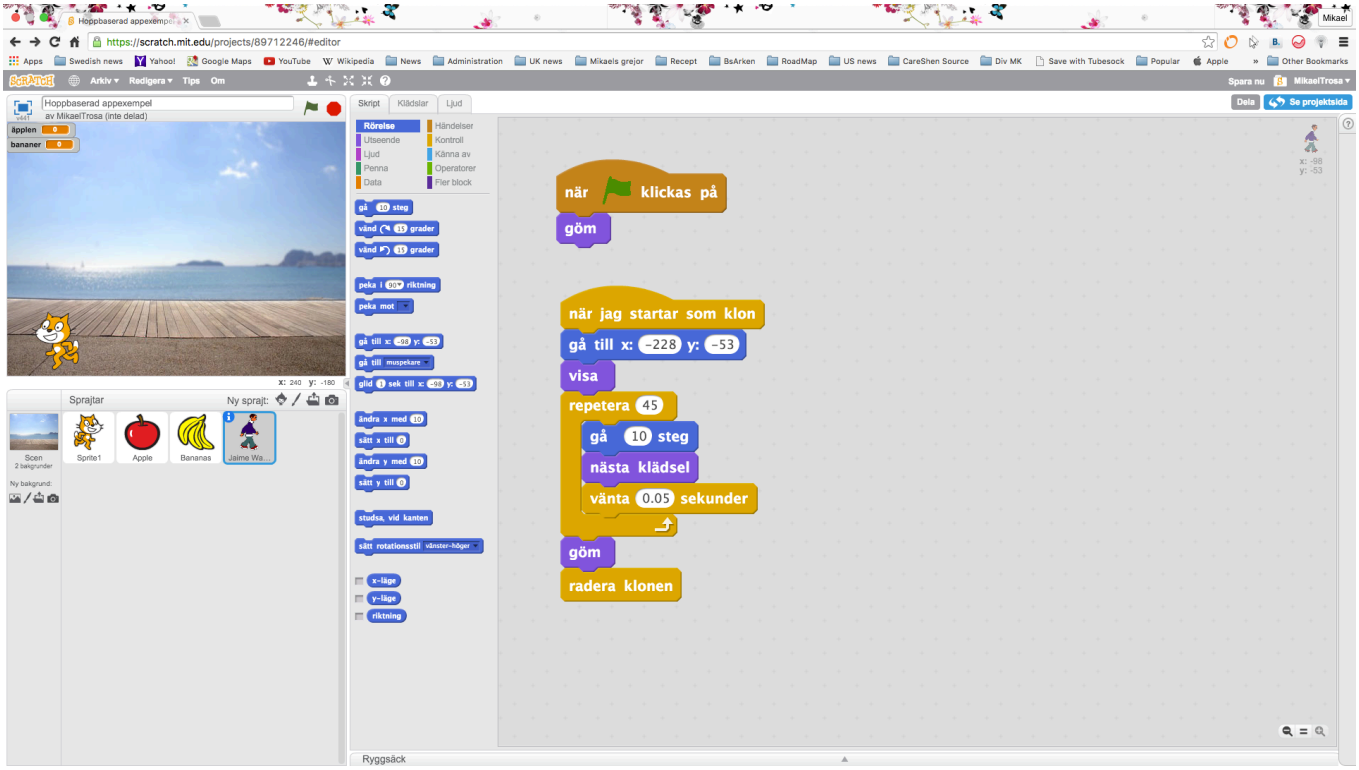
Äpplet



Bananen



Jaime Walking



Spelet laddas ner från http://bredbandarna.se/spel/Kattenhoppar.sb2. Gå till Arkiv på Scratchmenyn och välj "Ladda upp från din dator" och välj filen du just laddat ned. Du kan också använda följande länk: https://scratch.mit.edu/projects/92554991/