T2 L4: Bit.Trip-spring!

(Termin 2 Lektion 4)

I denna lektion så ska vi göra ett spel då man styr en liten ninja som kan hoppa jättehögt och åka igenom väggar! Det kommer se ut som att ninjan springer åt höger men den spriten kommer stå still och istället ska vi låta bakgrunden åka åt vänster!

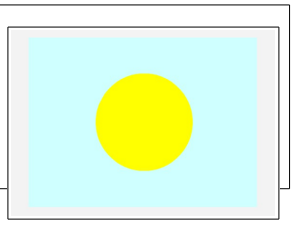
Koncept i fokus: ”animering”, ”x-värde”, ”y-värde”, ”lager” och ”funktioner”

Delmoment 1: Sätta upp scen, mark och ninjan

Vi börjar med att göra bakgrunden, rita mark och rita ninjans alla klädslar. Ninjan kommer behöva 4 klädslar till att springa, 1 till att hoppa och 1 till att sparka.

1. Skapa ett nytt Scratch-projekt och **radera** katten.

2. Rita en ny **bakgrund** och fyll den med färgen blå, som himmelen. Om du vill så kan du rita en stor sol.

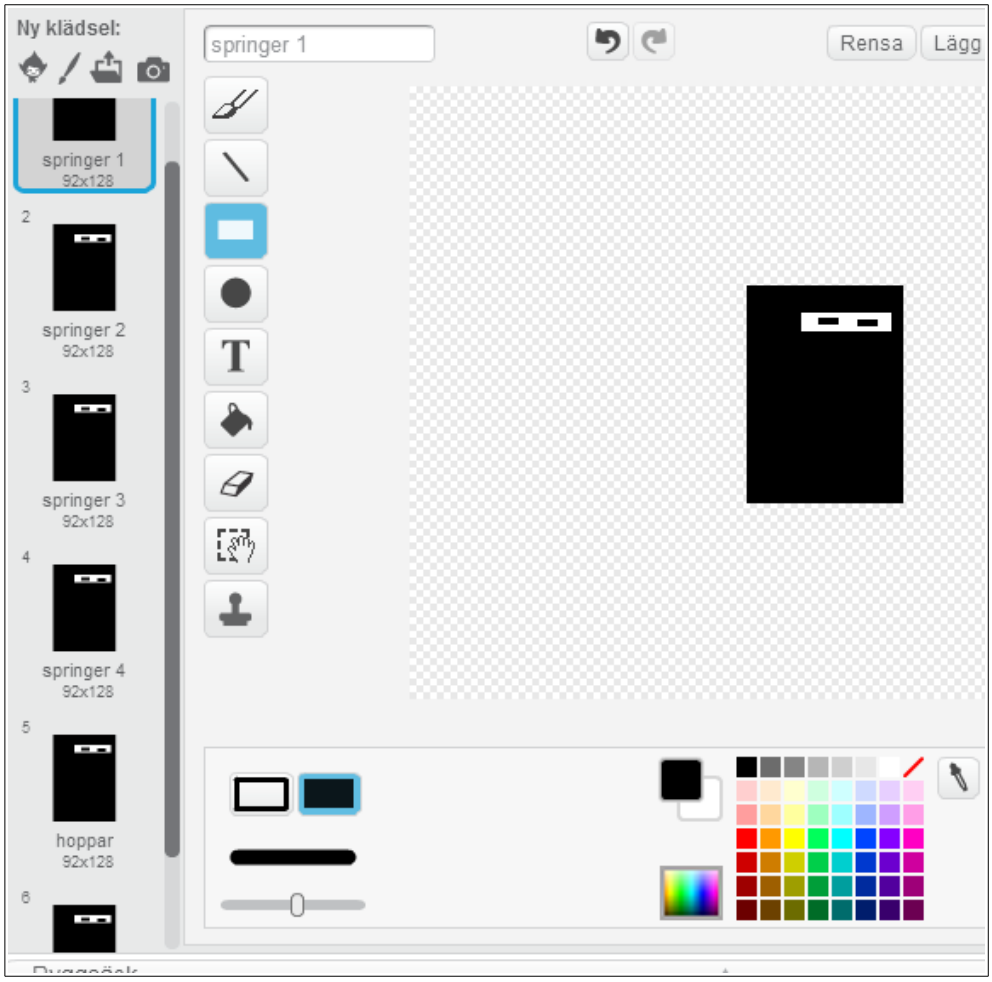


3. Rita en ny **sprite** och använd rektangelverktyget för att göra en rektangel på botten av bilden. Det är på den som våran ninja kommer springa.

4. Placera den så att den täcker botten av spel-skärmen och ge

spriten namnet **”mark”**.

5. Rita en ny **sprite**. Använd rektangelverktyget för att göra en rektangel och ett litet ansikte. Det blir en fyrkantig ninja. Sen så kopierar du klädseln så att vi har 6 såna. Sen ska vi rita olika ben på dem!



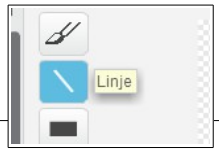
6. Ge **klädslarna** som vi just ritat namnet **”springer1”**, **”springer2”**, **”springer3”**, **”springer4”**, **”hoppar”** och **”sparkar”**.

7. Ge spriten namnet **”ninja”** så vi håller lite ordning på det hela.

Nu ska rita en animation för ninjans ben. Vi ska rita springande ben på

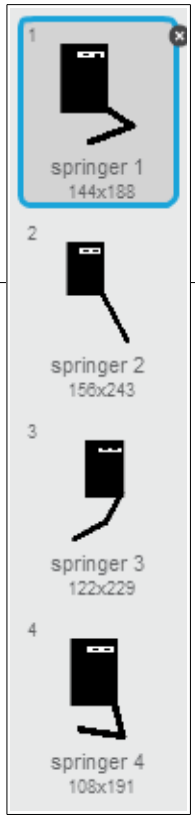
dem **4 första klädslarna.**

Vi börjar med det högra benet först! Du kan använda **linjeverktyget** så blir det raka linjer.



Du kan även ändra **tjockleken** på linjerna nere till vänster.

8. Först så har man **knät högt uppe** när man börjar springa. Sen har man **rakt ben** då man nuddar marken. Sen när man har tryck ifrån så har man **lyft benet bakåt.** Sist så drar man **fram benet igen** så att man kan ta ett nytt steg med det benet.



För att se att våran animation ser bra ut så ska vi göra skriptet för animationen.

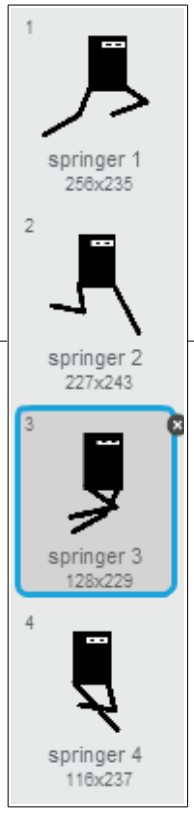
9. Skapa ett skript för spriten ”ninja” som:

- Startar när man klickar på flaggan
- För alltid:
    - vänta **0.2** sekunder
    - byt klädsel till ”springer 1”
    - vänta **0.2** sekunder
    - byt klädsel till ”springer 2”
    - vänta **0.2** sekunder
    - byt klädsel till ”springer 3”
    - vänta **0.2** sekunder
    - byt klädsel till ”springer 4”

Vi kan inte använda blocket ”nästa klädsel” för då kommer ”hoppar” och ”sparkar” att användas i spring-animationen.

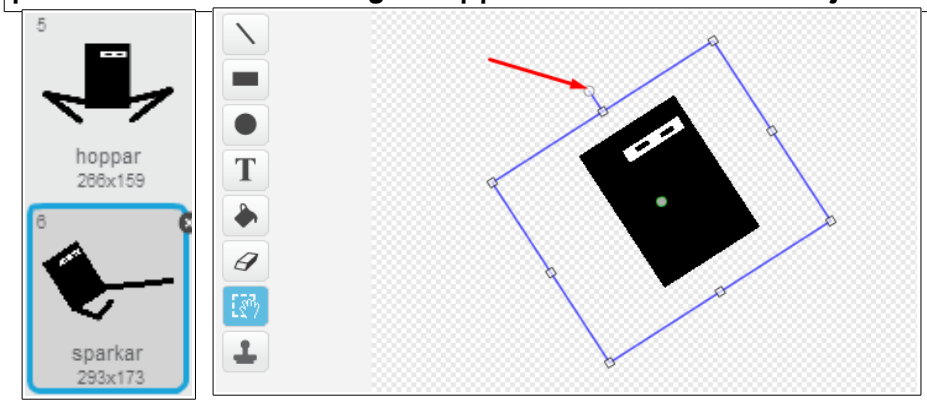
Nu kan vi se hur våran animation ser ut!
0.2 sekunder är **mindre än 1 sekund!** Pröva att ändra det till **0.5** för att se om det tar längre tid för ninjan att springa!

10. Nu ritar vi andra benet. Det är viktigt att andra benet är i marken när det första benet är i luften, och vice versa. Så på klädseln **”springer3”** så förbereder vi det andra benet att ta i marken. Sen så ritar vi samma ben igen som för andra benet, för man gör ju samma sak! Fast inte samtidigt.



Ser det ut som att ninjan springer? Det gör inget om det inte ser perfekt ut.

11. Nu ska vi rita två ben på när ninjan **hoppar** och två ben på när ninjan **sparkar**. Man kan använda **markeringsverktyget** för att markera ninjan och sen klicka
på bollen som är högst upp för att snurra ninjan.



Snyggt jobbat! Nu ska vi bara lägga till 2 sprites till.

12. Lägg till spriten **Saker/Marble Building** och ge den namnet **”bakgrundshus”**. Den ska åka sakta i bakgrunden så att det ser ut som att ninjan springer.

13. Lägg till spriten **”Saker/Rocks”** och ge den namnet **”hinder”**.

14. Lägg till en ny klädsel för **”hinder”**, välj **”Letters/I-Pixel”**.

Det är bokstaven i men vi ska använda det som en vägg.

Nu har vi alla sprites och klädslar som vi behöver!

Delmoment 2: Göra så att allting åker!

Vi vill att våran byggnad åker sakta åt vänster så att det ser ut som att den är långt borta. Vi vill också att vårat hinder ska åka till vänster, men den ska åka snabbare så att det känns som att den är närmare.

15. Flytta byggnaden så att den nedre kanten är precis förbi marken. Sen kollar du på byggnadens **Y-värde**. Det här värdet ska du skriva in i nästa skript.

Vi vill att byggnaden ska åka från höger till vänster, så vi vill att byggnadens **X-värde** ska ändras från **240** som är längst till höger, till **-240** som är längst till vänster.

16. Skapa ett skript för byggnaden som:

- Startar när spelet startar
- För alltid:
  - Gå till **x:240 y:** (skriv in ditt egna y:värde som du listade ut i punkt 14)
  - Glid 30 sekunder till x: -240 y: (skriv in ditt egna y:värde som du listade ut i punkt 14)

Om du klickar på skriptet, far byggnaden sakta åt vänster och sen dyker upp igen till höger för att åka igen till vänster?

Vi vill också att byggnaden ska vara i bakgrunden, så vi vill att den ska vara i **lagret** som är längst bak. Bakom våran ninja och framför våran sol.

17. Lägg till blocket **"gå tillbaka 3 lager"** innan för-alltid-loopen.

Är den nu bakom hindret och ninjan när spelet startar?

Snyggt! Ser det ut som att ninjan springer förbi byggnaden?

Nu ska vi göra så att vårat hinder också åker!

18. Vi vill att hindret ska börja längst till vänster med samma **x:värde** som byggnaden. Men vi vill att den ska vara lite längre ner så vi vill att **y:värdet** ska vara lite lägre.

19. Gå in på spriten **"hinder"** och klicka på klädseln "rocks" så att man ser stenarna på spel-skärmen. Placera spriten så att den är på marken så att ninjan kan hoppa över den. Märk vilket **y-värde** som

spriten ligger på så att du kan skriva in det i nästa skript för stenen.

20. Skapa ett skript för ”hinder” som:

* Startar när spelet startar
* gå till **x:240 y:** (skriv in ditt egna y:värde som du listade ut i punkt 18)
* **glid 5** sekunder till x:-240 y: (skriv in ditt egna y:värde som du listade ut i punkt 18)

När du startar spelet, far stenen lite snabbare än huset så att det ser ut som att man springer förbi den?

Snyggt!

Delmoment 3: Slumpmässiga hinder och tillstånd

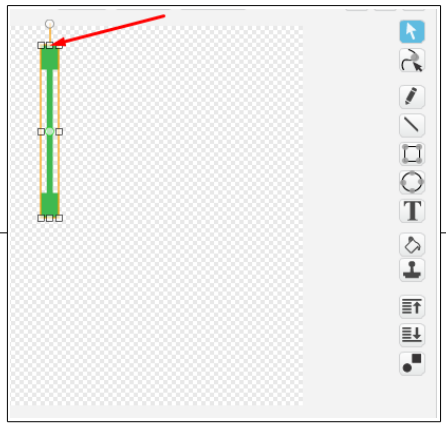
Vi ska nu göra så att hindret byter klädsel till en slumpmässig klädsel och sen ska vi göra så att ninjan kan hoppa och sparka igenom väggar.

21. Ändra i skriptet för ”hinder” så att det också byter klädsel till ett slumptal 1 till 2. Du kan även lägga till att hindret göms när den glidit färdigt och visas efter att den gått till högerkanten igen.

Är det nu slumpmässigt vilken klädsel som ”hinder” använder när den åker till vänster?

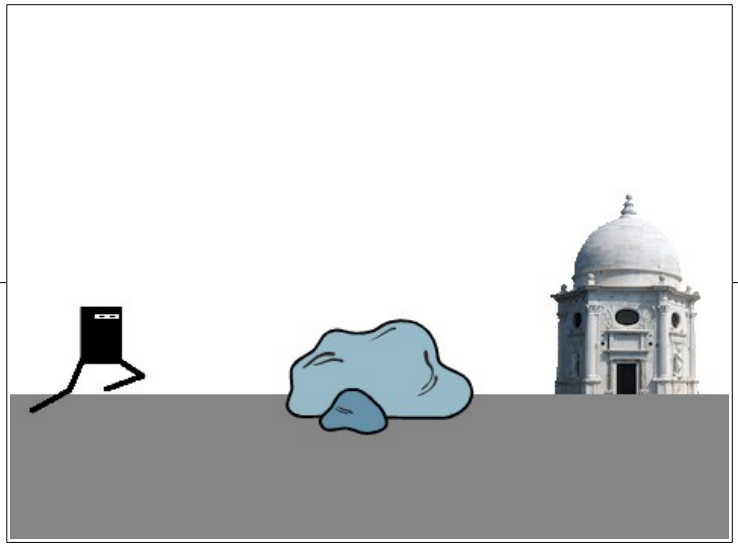
Nu ska vi göra i:et så att det ser mer ut som en vägg.

Markera i:et med **markeringsverktyget** och dra i den mittersta översta lilla **kvadraten** så att du kan göra så att i:et nästan når toppen av bildytan.



Nu har vi en hög vägg!

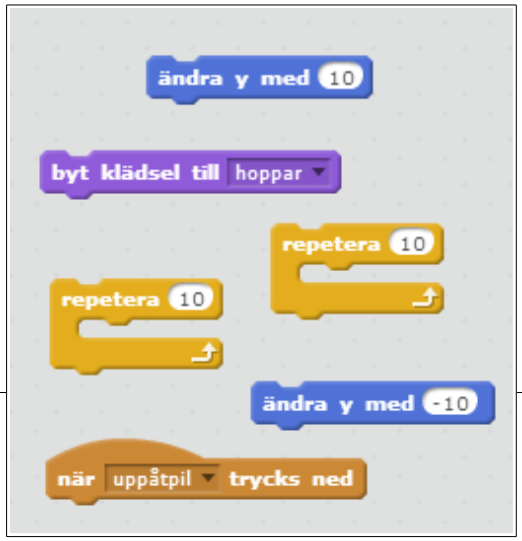
22. Ändra storleken på ninjan med **förminskningsverktyget** så att ninjan är en passande storlek. **Placera** ninjan till vänster i spelet så att spelaren får lite tid att reagera på hinder.



Nu ska vi göra så ninjan kan hoppa. Vi gör så att **y-värdet** ökar och sen sänks igen. Då far ninjan upp, sen ner.

23. Skapa ett skript för **"ninja"** som:

- Startar när **"uppåtpil"** trycks ner
- byt klädsel till **"hoppar"**
- repetera **10** gånger:
  ändra **y med 10**
- repetera **10** gånger:
  ändra **y med -10**



Vi vill att ninjan ska kunna hoppa över
stenen men inte över det **gröna i:et**. På det sättet så kan vi göra så att spelaren måste sparka när i:et kommer och hoppa när stenen kommer.

24. **Nu** kan du ändra värdena **10** och **-10** så att ninjan kan hoppa högre. **Du** kan också testa att **ändra hur många repetitioner** det ska vara. **Se** bara till att ninjan hamnar på samma ställe som den startade på.

Y-värdet ska vara samma efter att ninjan hoppat som innan ninjan hoppat! Det gör du genom att alltid ha samma antal repetitioner på båda looparna och samma ändring på y-värdet, men negativt i den under loopen.

Jag valde att ha **30** repetitioner som ändrar värdet på y: **4** och sen **-4** varje gång. Då kan min gubbe hoppa över stenen men hoppar inte över i:et.

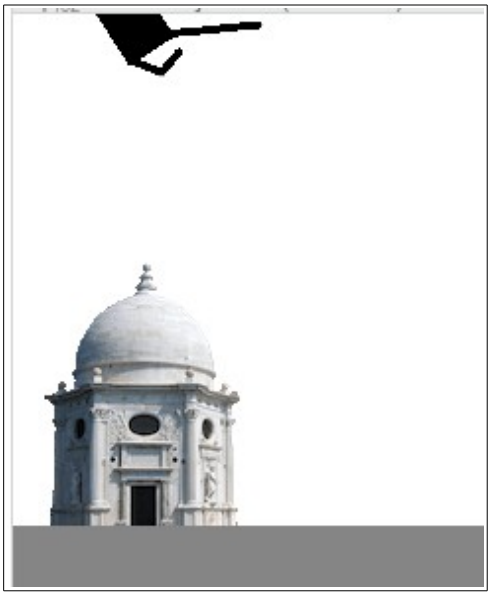
Nu ska vi göra så att ninjan byter klädsel till **"sparkar"** och fortsätter sparka i **1** sekund, när man trycker på pil höger.

25. **Skapa** ett skript i spriten "ninja" som:
* **S**tartar när **"högerpil"** trycks ner
* byt klädsel till **"sparkar"**
* vänta 1 sekunder

Nu när man trycker ner **"högerpil"** så byter ninjan till rätt klädsel, men vårat skript för att animera att ninjan springer byter klädsel direkt och gör så att ninjan inte kan sparka i en hel sekund.

När ninjan hoppar och sparkar så stannar inte animationen på dem bilderna. Spring-animationen tar över. Så lägg till blocket ”stoppa andra skript i sprite” så att man kan hoppa och sparka.

Nu får vi dock **2** nya buggar. Pröva att hoppa och i luften sparka och sen hoppa **igen**. Vad händer?

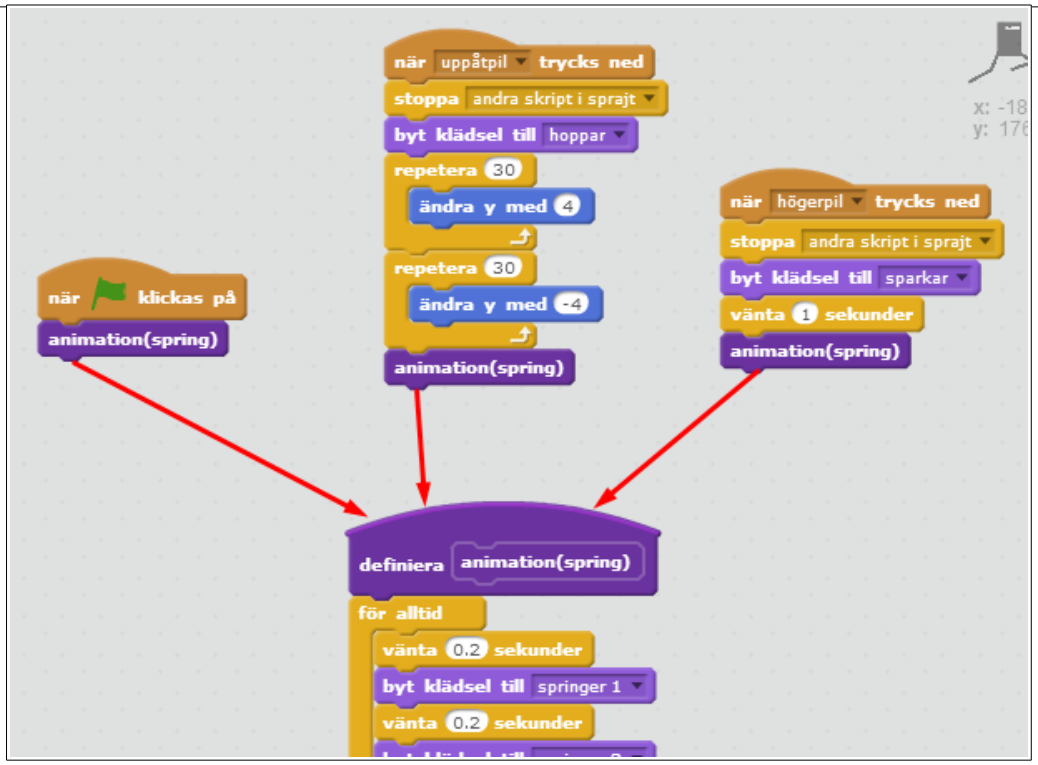


Det andra problemet är att ninjan aldrig börjar **animationen för att springa**. Så man kan bara hoppa och sparka nu.

Så nu för att lösa det ena problemet med att ninjan inte springer så ska vi kopiera koden för animationen, sätta in den i en funktion och sen göra så att funktionen körs i ninjans alla tre skript.

26. Kopiera koden som gör att ninjan springer och sätt den under ett funktions-block. Nämn funktionen till **”animation(spring)”**

Sen sätter du funktionsblock under alla tre skript så att dem alltid sätter igång animationen för att springa.



Om ninjan inte går tillbaka till sin position så lägg till kod så att spriten går till rätt **koordinater** när man klickar på flaggan. Mina koordinater blev x:-197 y: -41 men du kan ha andra värden på x och y!

Nu dök en ny bugg upp! Kan du trycka på högerpil och göra flera sparkar i rad? Eller kan du göra flera hopp i rad? Vad tror du det beror på?

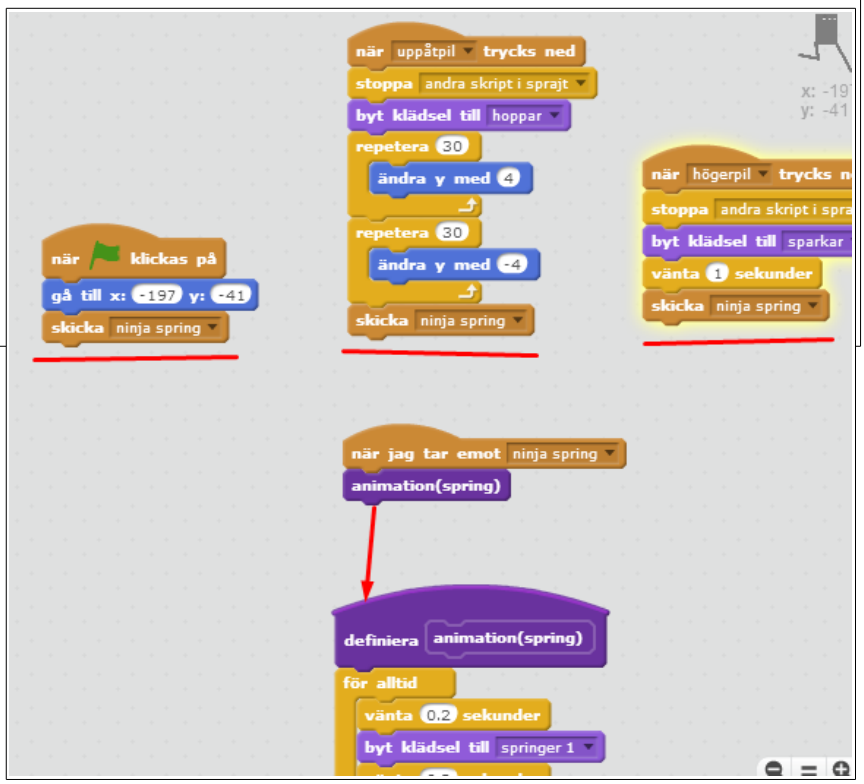
Det är nämligen så att alla tre skript fastnar i **animation(spring)** efter att dem antingen har sparkat eller hoppat. Sen så kan vi inte köra samma skript igen. Vi ska lägga till att vi istället skickar ett

meddelande så att ett separat skript körs som sen kör **animation(spring).**

27. Vi har redan gjort en funktion som gör att ninjan springer. Ändra på de tre skripten så att de inte kör funktionen, utan istället skickar meddelandet **”ninja spring”.**

28. Skapa nu ett till skript i ninjan som:
* Startar när den tar emot meddelandet **”ninja spring”**
* kör funktionen **”animation(spring)”**

Nu fastnar ingen av skripten i loopen, utan ett separat skript körs, så nu kan vi hoppa och sparka hur mycket vi vill!



Om man vill så kan man ta bort funktionen och bara låta den koden köras direkt under **när jag tar emot spring.** Men jag låter funktionen vara kvar. Det kan hända att man behöver den sen!

Nu har vi fixat problemet med att ninjan inte fortsatte spring-animationen!

Nu ska vi fixa så att man inte kan hoppa och sparka samtidigt.

29. Skapa en variabel i ninjan (för alla sprites) som heter **”tillstånd ninja”**. Den ska vi ha så att ninjan och dem andra spritarna kan få veta om ninjan sparkar, hoppar eller bara springer.

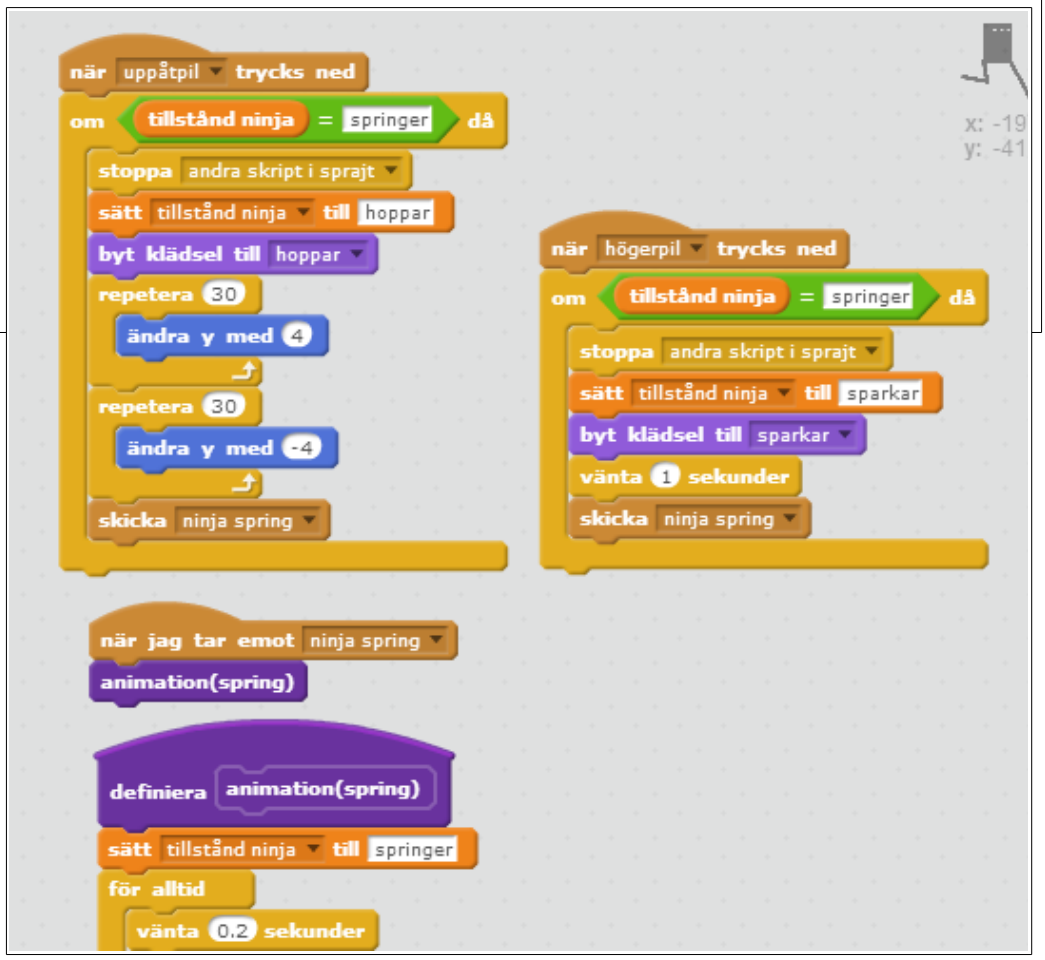
30. I ninjans skript som ska starta spring-animationen, precis innan så lägg till ett block så att variabeln **”tillstånd ninja”** sätts till **”springer”**.

31. Ändra nu skriptet i ninjan som startar när man trycker ner uppåtpil så att det bara kör koden om tillstånd ninja är **”springer”** och direkt sätter tillstånd till **”hoppar”** eller **”sparkar”**. Gör det genom att sätta all koden under skripten i Om-då block.

Så här ser min kod ut.

Din kod kan se lite annorlunda ut.

Det viktiga är att koden gör det vi vill.



Fungerar det nu att hoppa och sparka men inte samtidigt?

Om du visar variabel **”tillstånd ninja”** så kan man se att vad variabeln har för värde när man hoppar, sparkar eller springer.

När man kodar så är det ganska skönt att se en del variablers värden, då kan man lätt hitta och lösa **buggar**.

Delmoment 4: Farliga hinder och fall-animation

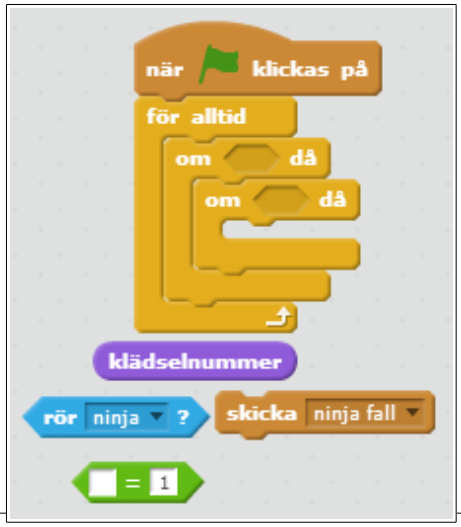
Om man rör vid stenen så vill vi att ninjan faller. Det samma vill vi ska hända när man rör vid väggen, men om man sparkar så ska väggen falla istället. Vi ska lägga till att ninjan kan ta emot ett fall-meddelande och ett skript i väggen som kollar om man sparkar när man rör den.

Vi börjar med att fokusera på **hindret**. Vi ska göra så att när hindret är en **sten** så ska en viss kod köras, och när hindret är en **vägg** så ska en annan kod köras.

32. Skapa ett skript i **hindret** som:

- Startar när man klickar på flaggan
- för alltid:
    - om rör ”ninja”:
        - om klädselnummer = 1:
            - skicka ”ninja fall”

Det är ett om-block i ett om-block i en för-alltid-loop!



Om vi ville så hade vi kunnat göra som med ninjan och skapa en variabel som heter **”hinder tillstånd”**. Den hade kunna hålla koll på vilket hinder den just nu va. Vi behöver inte göra det nu för vi kan använda klädselnummer istället och spara lite tid. Om hinder använder klädseln med stenen så är det ju en sten. Ser du någon nackdel med att använda klädselnummer istället för tillstånd? Tillståndet kunde vi ge namn som ”hoppar” men klädselnummer är alltid ett nummer.

Kanske kunde man lösa det på ett **tredje** sätt också? Det finns nästan alltid flera olika lösningar. Då kan man fundera på vilken lösning som passar bäst just då!

För att se att hindrets skript fungerar bra, så ska vi göra ett enkelt skript för ninjan då den faller.

33. Skapa ett skript i **”ninja”** som:
* Startar när den tar emot meddelandet ”ninja fall”
* stoppar andra skript i sprite
* repetera tills rör kant
    vänd åt **höger** **30** grader
    ändra **y** med **50**
* göm

34. Eftersom vi använder **göm** och gör så att ninjan vänder några **grader** så måste vi lägga till att ninjan **visas** och **pekat i 90 riktning** i skriptet i ninjan som börjar när spelet börjar.

Flyger ninjan iväg när han springer på stenen?

Nu ska vi lägga till så att väggen också kan interagera med ninjan.

Vi börjar med att göra en animation för väggen så att den kan gå sönder om ninjan sparkar den.

35. Skapa en **funktion** i **”hinder”** som du kallar **”fall(vägg)”**:
* säg ”krasch!” i 1 sekunder

36. Ändra i skriptet i ”hinder” som kollar om hinder rör ninjan så att:
* För alltid:
    om rör ”ninja”:
        om klädselnummer = 1:
            skicka ”ninja fall”
        om klädselnummer = 2:
            om ”tillstånd ninja” = ”sparkar”:
                kör funktionen **”fall(vägg)”**
            annars:
                skicka ”ninja fall”

Fungerar det nu att hoppa över stenen och klara sig igenom väggen?

Nu är **prototypen** för spelet klart! **Prototyp** är en ofärdig sak som bara gör det mest grundläggande, för att se om det går att göra.

Fungerar ditt spel som du vill?

När man gjort en **prototyp** så kan man sen förbättra den!

Delmoment 4: Förbättra prototypen

Nu får du möjlighet att lägga till mer till spelet! Välj själv vilka du vill pröva! Om du kommer på en egen idé så kan du göra den!

A. Du kan förbättra spelet genom att ändra i funktionen ”fall(vägg) så

att väggen kanske vänder sig och lägger sig på marken? Eller att den går sönder genom att byta klädsel till andra klädslar.

B. Du kan ge ninjan fler tillstånd, kanske ”glida under” som gör att ninjan glider under ett hinder. Då kan man lägga till ett hinder som är som stenen men lite högre, så att ninjan måste glida under den!

C. Du kan lägga till ljud när ninjan hoppar och när ninjan sparkar på hindret! Jag lade till ljudet **”Effekter/water drop”** när ninjan hoppade, det kan du göra om du vill!

D. Du kan lägga till att efter att ninjan fallit så kan man trycka på ”mellanslag” för att snabbt börja ett nytt spel.

37. Sätt spelet i fullskärm och visa det för en vuxen!
    Berätta vad du gjorde för förbättringar på spelet!
    Berätta vad dina funktioner gör!

Spelet ska heta **T2 L4** så att det är lätt att hitta!
Spara!
(T2L4 är en förkortning av Termin 2 Lektion 4)
När man kodar så är det vanligt att ha förkortningar så att man inte behöver skriva så mycket. Ibland kan det skapa förvirring, så det gäller att veta vad som betyder vad!

Nu är du klar med hela lektion T2 L4 ! Nu kan du börja på lektion 5 (T2 L5) !