Første Program

Tast følgende eksempel ind i Processing editoren og tryk på -knappen:

```
size(400, 400)

# Træ
rect(55, 50, 10, 20)
ellipse(60, 35, 30, 40)
```

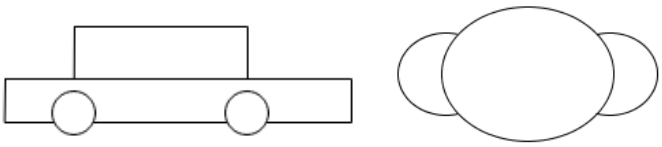
Tilføj derefter:

```
# Kraftværk
rect(120, 50, 60, 30)
rect(160, 20, 10, 30)
triangle(120, 50, 136, 40, 136, 50)
triangle(136, 50, 152, 40, 152, 50)
```

Og:

```
# Vindmølle
line(300, 50, 320, 51)
line(300, 50, 289, 67)
line(300, 50, 291, 32)
line(300, 50, 300, 90)
```

Prøv nu at tegne en bil og en sky:



Farver

I skal nu farvelægge figurerne. Til det skal I bruge `fill(r, g, b)`-funktionen, der vælger hvilken farve der skal bruges til udfyldning og tekstfarve. Det handler om at kalde `fill` de rigtige steder! Som argumenter angiver man mængden af rød (0-255), blå (0-255) og grøn (0-255).

Her er nogle grundfarver:

```
fill(255, 0, 0)  # rød
fill(0, 255, 0)  # grøn
fill(0, 0, 255)  # blå
```

```
fill(0, 0, 0)       # sort
fill(255, 255, 255) # hvid
fill(255, 255, 0)   # gul
```

Find eventuelt farver ved hjælp af en online farvevælger eller RGB farve tabel. Søg for eksempel efter "rgb color picker".

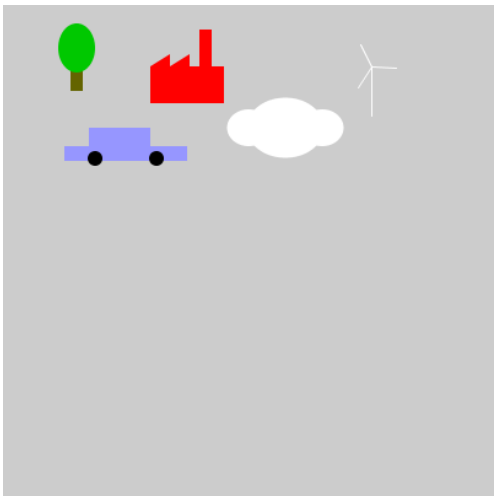
Streger og omrids:

Til at angive farven på streger (fx `line`) og omrids bruges `stroke(r, g, b)`. Prøv også funktionen `noStroke()`, til at slå optegning af omrids fra.

Martin Dybdal & modified by DS
ds@di.ku.dk

Eksempel

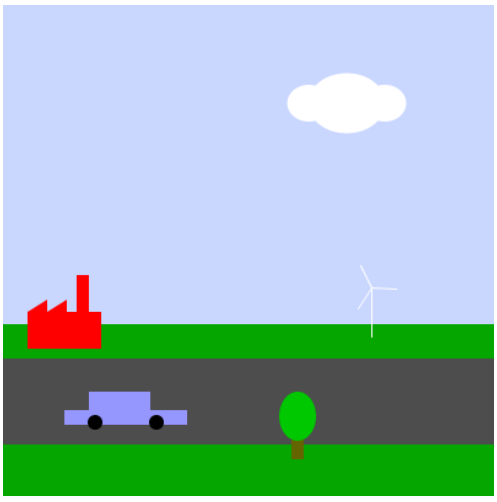
Her er et eksempel på hvordan det kan se ud efter farvelægning:



Green City

Brug det I har lært til at få det til at blive en lidt pænere scene, med baggrund, forgrund og de ekstra detaljer som I synes der skal være der. Jeg har fx tegnet en vej og flyttet lidt rundt på figurerne.

Tip: For at ændre baggrundsfarven fra grå kan i bruge `rect` som i allerede kender, men i kan også bruge kommandoen `background(r, g, b)`. Den sletter alt og udfylder skærmen med den angivne farve.



Husk at bruge kommentarer, så I nemt kan finde rundt i koden!

Martin Dybdal & modified by DS
ds@di.ku.dk

Variable

Opret et nyt projekt ("File" -> "New") og gem med det samme projektet. Kald det "Akvarie".
Skriv denne stump kode ind:

```
size(400, 400)
fishX = 150
ellipse(fishX, 200, 120, 75)
triangle(fishX - 60, 200, fishX - 90, 170, fishX - 90, 230)
```

Prøv at ændre 150 til et andet tal i angivelsen af `fishX`.

Tilføj nu følgende:

```
eyeSize = 15
ellipse(fishX + 30, 190, eyeSize, eyeSize)
```

Prøv at ændre på værdien af `eyeSize`.

Opgaver:

I har nu tilføjet en fisk der kan flyttes, bare ved at ændre én værdi.

- Farvelæg fisken
- Giv fisken en finne, som flytter med når I ændrer `fishX`.
- Giv fisken en pupil, som flytter med når I ændrer `fishX`.
- Lav en ny variabel, `fishY`, der styrer fiskens y-position

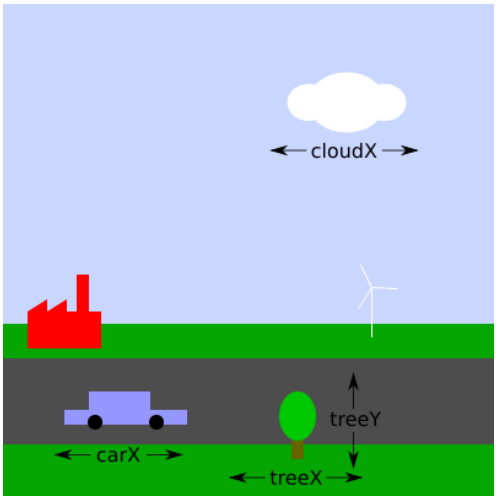
Husk at gemme projektet. Vi skal arbejde videre med det senere. missing link

Martin Dybdal & modified by DS
ds@di.ku.dk

Green City forstat

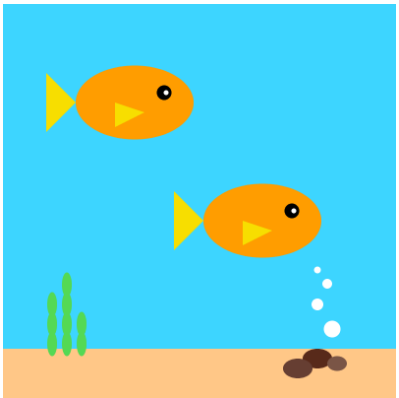
Skift over til elbil-projektet og indfør variable til angivelse af objekternes placering, så vi senere kan animere objekterne.

- Lav en variabel `carX` så bilen kan bevæge sig frem og tilbage
- Lav en variabel `cloudX` så skyen kan bevæge sig frem og tilbage
- Lav en variabel `treeX` så træet kan flyttes horizontalt
- Lav en variabel `treeY` så træet kan flyttes vertikalt



Akvarie fortsat

Brug nu det I har lært til at udvide akvarie projektet, her er et eksempel, men brug gerne jeres fantasi! Der er fx brugt en variabel til x-koordinat af tangplanten, og en anden variabel til x-koordinat af hele gruppen af sten (som en enhed). Der er oprettet et ekstra sæt variable `fish2X`/`fish2Y` til at styre placeringen af den ekstra fisk.



Martin Dybdal & modified by DS
ds@di.ku.dk

Funktioner

Funktioner giver mulighed for at genbruge den samme kode flere steder, navngive hele blokke af kode og sætte struktur på kode.

Åbn akvarie-projektet. Tilføj følgende "fiske-tegne-funktion" nederst i projektet. BEMÆRK! Linjeindrykning med 4 mellemrum er vigtig!

```
def drawSimpleFish(x, y):
    ellipse(x, y, 120, 75)
    triangle(x - 60, y, x - 90, y - 30, x - 90, y + 30)

drawSimpleFish(100,  50)
drawSimpleFish(300, 200)
drawSimpleFish(20, 20)
drawSimpleFish(80, 80)
```

Nu går det meget hurtigere med at få fyldt akvariet med fisk, og vi undgår at kopiere kode.

Opgaver:

Opret jeres egen `drawFish(x, y)` funktion, der tegner hele jeres fisk med farve, finner og øjne.
beginexercisebox[adjusted title= Green City forstat] Ovre i elbil-projektet kan I også prøve at skrive en funktion til at tegne træer:

```
def drawTree(treeX):
    fill(100, 100, 0)
    rect(treeX - 5, 350, 10, 20)
    fill(0, 200, 0)
    ellipse(treeX, 335, 40, 50)

drawTree(160)
drawTree(300)
```

Få sat struktur på koden til elbil-projektet ved hjælp af funktioner:

- Skriv en `drawCloud(x)`-funktion der tegner en sky
- Udvid `drawTree(x)`-funktionen til at også tage imod et y-koordinat
- Skriv en `drawCar(x)`-funktion der tegner en bil
- Skriv en `drawPowerplant()`-funktion og en `drawWindmill()`-funktion, der tegner hhv. kraftværket og vindmøllen. Vi får ikke behov for at flytte på dem, så de behøver ikke tage koordinater som argument.

Kald alle funktionerne nederst i dit program. For eksempel:

```
drawTree(150, 235)
drawTree(240, 335)
drawPowerplant()
drawWindmill()
drawCar(50)
drawCloud(280)
```


Martin Dybdal & modified by DS
ds@di.ku.dk

Animation og funktionerne

Opret et nyt midlertidigt projekt (I behøver ikke gemme det). Tast denne stump kode ind:

```
x = 50

def setup():
    size(400, 400)

def draw():
    global x
    background(255, 255, 255)
    fill(255, 0, 0)
    ellipse(x, 100, 30, 30)
    x = x + 1
```

Funktionen `draw` kaldes automatisk 60 gange i sekundet!

Opgaver:

- Prøv at ændre 50 til et andet tal
- Prøv at ændre linjen `x = x + 1` til `x = x - 1` eller til `x = x + 5`
- Forsøg at flytte kaldet til `background` fra `draw` til `setup` - hvad sker der?

BEMÆRK!!:

Når man bruger `setup/draw` er det ikke tilladt at kalde tegne-funktioner udenfor `setup` og `draw`, alt skal flyttes ind i de to funktioner.

Akvarie fortsat

- *Omskrivning af akvarieprojektet til brug af* `setup/draw`*:*
  - Tilføj tomme `setup` og `draw`-funktioner nederst i programmet
  - Kald `size(400, 400)` i `setup`
  - Kald alle tegnefunktionerne i `draw`, inkl. tegning af baggrunden
- *Få fiskene til at svømme:*
  - Opret to globale variabler `fish1X` og `fish2X` (før `setup/draw`)
  - Brug de nye variable som x-argument når I kalder `drawFish()`
  - *HUSK* linjerne: `global fish1X` og `global fish2X`
  - Opdater variablerne med $+1/-1$ inde i `draw`-funktionen

Martin Dybdal & modified by DS
ds@di.ku.dk

Green City forstat

- Opret to globale variabler `carX` og `cloudX`
- Få bilen til at køre mod højre
- Få skyen til at starte uden for billedet i højre side og bevæge sig mod venstre

Tilfældighed

Opret et helt nyt projekt, gem det som "random_circles". Tilføj følgende kode:

```
def setup():
    size(400, 400)

def draw():
    x = random(0, width)
    y = random(0, height)
    ellipse(x, y, 30, 30)
```

Opgaver:

- Få cirklerne til at ændre størrelse tilfældigt
- Få cirklerne til at blive tegnet i tilfældige farver. Eksperimenter jer frem til en farveskala I synes er pæn.

Input fra musen

Placeringen af musen kan aflæses med variablerne `mouseX` og `mouseY`. En slags tegneprogram kan skrives således:

```
def setup():
    size(800, 800)
    background(255, 255, 255)

def draw():
    fill(0, 0, 0)
    ellipse(mouseX, mouseY, 5, 5)
```


Martin Dybdal & modified by DS
ds@di.ku.dk

Tastatur input

Åbn tegneprograms-projektet og prøv at tilføje følgende nye funktion:

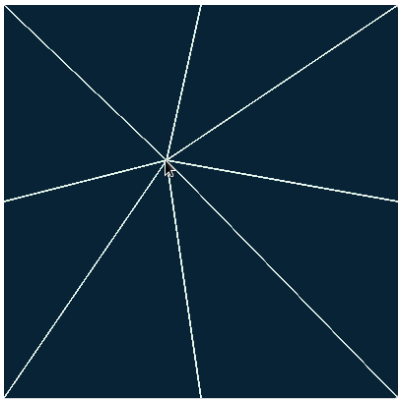
```
def keyPressed():
    background(255, 255, 255)
```

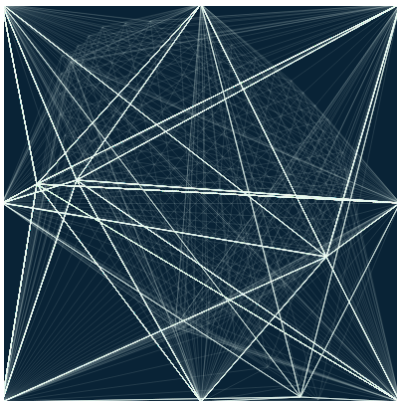
Start programmet og tryk på en vilkårlig tast på tastaturet. For at tjekke efter en specifik tast kan variablen "`key`" aflæses.

```
def keyPressed():
    if key == 'c':
        background(255, 255, 255)
```

Nyt projekt: Kreativt tegneprogram

Opgaven er nu at lave et tegneprogram der er mere kunstfærdigt. Tegn linjer fra alle hjørner og midtpunkter på siderne. Brug variablerne `width` og `height`.





Husk at gemme projektet!

Martin Dybdal & modified by DS
ds@di.ku.dk

Betingelser

Med betingelser kan vi få ting til at ske når specielle kriterier er opfyldt. Åbn akvarie-projektet og prøv at indsætte følgende i `draw`-funktionen:

```
if fish1X > 500:
    fish1X = -40

if fish1X < -50:
    fish1X = 450
```

Hvad sker der?

Skifte retning

Ved at lave en variabel der indeholder retningen som fisken svømmer, kan vi få den til at ændre retning når den når siderne.

- Definer en global variabel `fish1XVelocity` og sæt den til 1
- Ændr `fish1X = fish1X + 1` til `fish1X = fish1X + fish1XVelocity`
- Fjern den tidligere `fish1X`-betingelse
- Tilføj disse betingelser:

```
if fish1X > 400:
    fish1XVelocity = -1
if fish1X < 0:
    fish1XVelocity = 1
```

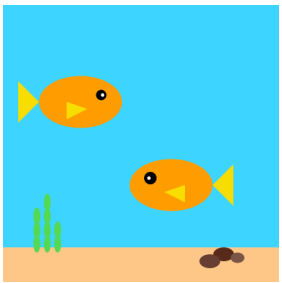
Ændre udseende

Prøv at ændre `drawFish`-funktionen, så den tager retningen af fisken med som argument, og brug en betingelse til at tegne finner og øjne forskelligt alt efter hvilken retning fisken svømmer.

```
def drawFish(x, y, eyeSize, velocity):
    ... tegn kroppen ...

    # Svømmer mod højre
    if velocity >= 0:
        ... tegn finner og øjne ...

    # Svømmer mod venstre
    if velocity < 0:
        ... tegn finner og øjne ...
```



Gør akvarieprojektet færdigt

Nu er vi sådan set færdig med akvarie-projektet. Tilføj eventuelt flere elementer. F.eks. bobler der dukker op fra bunden og svømmer mod overfladen.

Martin Dybdal & modified by DS
ds@di.ku.dk

Flere opgaver om betingelser

Åbn Green City projekt og gør følgende:

- Få skyen til at flytte tilbage til start og komme forbi igen og igen, hver gang den rammer kanten
- Få bilen til at vende om i begge ender.

Opgaver:

Bilen skal stoppe når batteriet er tomt

- Definer en global variabel `carBattery` og sæt den til 100
- Reducer den med 0.1 hver gang `draw` kaldes
- Vis batteriets status vha. `text(string, x, y)`-funktionen. Husk at konvertere tallet til en string vha. `str()`.
- Hvis batteriet er tomt (`<= 0`) skal bilen stoppe (sæt velocity til 0)

Tilføj en `keyPressed()` -funktion og gør så man kan oplade elbilen, når man trykker 'C':

- Læg `0.3` til `carBattery`, hver gang der tykkes 'C' på tasturet
- Hvis `carBattery` overstiger 100, så sæt den til 100, så man kan ikke lade til mere end 100%

Martin Dybdal & modified by DS
ds@di.ku.dk

Skftiende vindhastighed

Variablen `frameCount` tæller hvor mange gange `draw` er kørt siden programmet startede. Indtast dette i draw-funktionen i elbil-projektet:

```
fill(0, 0, 0)
text(frameCount, 350, 20)
if frameCount % 60 == 0:
    print(frameCount)
```

Eftersom `draw` udføres med *60 frames per second*, vil **`print`**-funktionen blive udført hvert sekund.

Det kan vi også bruge i elbilsprojektet, til at opdatere vindhastigheden hvert sekund.

- Opret en global windSpeed variabel
- Opdater windSpeed med en ny tilfældig værdi hvert sekund
- Vis vindhastigheden vha. `text()`
- Opret en global variabel `powerplantOn`
- Gør så man kan tænde og slukke kraftværket med en tryk på 'p'
- Tegn røgskyer over kraftværket når det er tændt
- Opret en variabel `co2emission` og læg lidt til så længe kraftværket er tændt. Vis værdien med `text()`

Martin Dybdal & modified by DS
ds@di.ku.dk

Lister og for loops (Optional)

Tag et kig på eksemplet nedenfor,

```
bubbleColor = color(156, 237, 250)#, bubbleOpacity)
seaColor = color(48, 213, 200)
lst = range(1,400,50)
lst2 = [(20,20,20), (50,50,50), (100,100,100)]
circle_list = []
xs = [20, 50, 150]
ys = [20, 50, 150]

def setup():
    size(400,400)

def drawCircle(x,y):
    print(x)
    stroke(255,255,255)
    noStroke()
    fill(bubbleColor)
    circle(x, y, 20)

def drawCircle2(x):
    stroke(255,255,255)
    noStroke()
    fill(bubbleColor)
    circle(x, 200, 50)

def createCircleList(n):
    for i in range(n):
        circle_list.append(i*60)

def draw():
    background(seaColor)
    createCircleList(10)

    for i in circle_list:
        drawCircle2(i)
    for i in range(3):
        drawCircle(xs[i], ys[i])
```

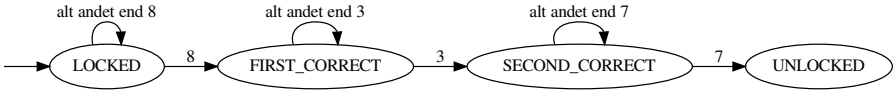
- .

Prøv at bruge loops og lister til at enten bevæge jeres fisk eller lave flere fisk i forskellige lokationer.

Martin Dybdal & modified by DS
ds@di.ku.dk

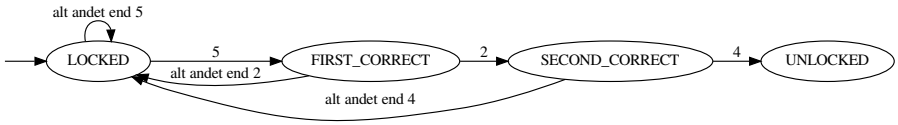
Tilstandsmaskiner (Finite state machines)

Som det første eksempel på en tilstandsmaskine skal vi lave en elektronisk kombinationslås, som dem der bruges til adgangskontrol på døre.
Tilstandsdiagram:



Hent Processing.py koden til kombinationslåsen her: `http://kortlink.dk/ufdh` og kopier den ind i et nyt Processing-projekt.

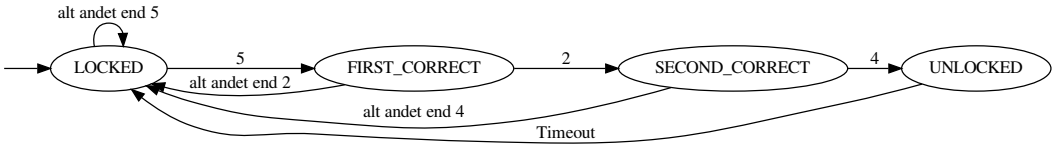
- Afprøv kombinationslåsen og følg med som tilstanden skifter. Hvis man ikke kendte løsenet (passwordet), hvor mange forsøg kræver det så at gætte sig frem?
- Ændr koden, så løsenet i stedet er 524
- Prøv at ændre koden til at følge dette diagram i stedet, hvor forkerte tryk sætter tilstanden tilbage til start:



- Tegn et udvidet tilstandsdiagram over kombinationslåsen, med en ekstra tilstand, så der kræves 4 cifre. Tilføj dernæst den ekstra tilstand i koden.

Automatisk genlås efter 2 sekunder

Lad os udvide låsen, så døren automatisk låser igen efter 2 sekunder, hvilket svarer til 120 frames:



- Opret en global variabel "`timer`" og sæt den til 0
- Sæt `timer`-variablen til 120 så snart låsen bliver låst op, det vil sige når den skifter `lockState` til `"UNLOCKED"`.
- Tæl ned med timeren i hver frame (tilføj følgende til `draw`-funktionen):

```
global timer
if timer > 0:
   timer = timer - 1
```

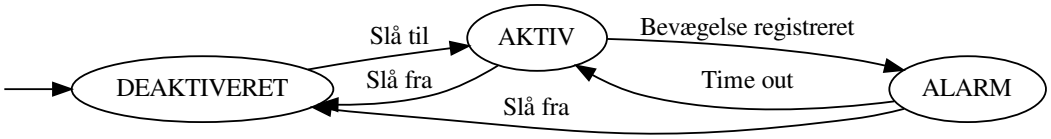
- Når timeren er talt helt ned, skal låsen åbnes. I `draw`-funktionen skal I nu tjekke om vi er i tilstanden `"UNLOCKED"` og timeren samtidig er talt ned til 0. Indsæt følgende i `draw`:

Martin Dybdal & modified by DS
ds@di.ku.dk

```
    if lockState == "UNLOCKED":
       if timer == 0:
           lockState = "LOCKED"
```

Alarm

Et andet eksempel på en tilstandsmaskine er en tyverialarm. Den har tre tilstande "DEAKTIVERET", "AKTIV" og "ALARM". Her er tilstandsdiagrammet:



Opret et nyt projekt og indtast koden herunder. Den indeholder funktionalitet til at opdage et indbrud, "bevægelse registreret" i figuren ovenfor. Det sker når man bevæger musen ind i cirklen i midten.

```
alarmState = "DEAKTIVERET"

def setup():
   size(400, 400)

def draw():
   global alarmState
   background(255, 255, 255)
   fill(0)
   text(alarmState, 20, 20)
   text("Slaa alarmen til/fra ved at trykke 's'", 20, 40)

   noFill()
   ellipse(200, 200, 100, 100)

   if alarmState == "AKTIV":
       # Tjek om musens cursor er inde i cirklen
       if dist(mouseX, mouseY, 200, 200) < 50:
           alarmState = "ALARM"
   elif alarmState == "ALARM":
       text("Indbrud", 20, 60)

def keyPressed():
   # indtast kode til at slå alarmen til/fra her
```

Det er nu din opgave at tilføje de sidste dele der gør det muligt at slå alarm til/fra og at alarmen stopper automatisk efter en timeout.

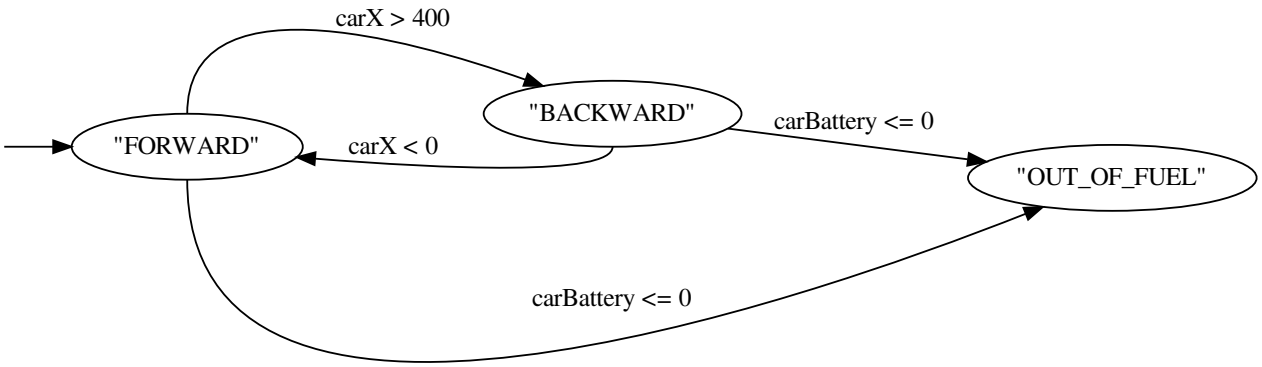
- *Slå til/fra:* Tilføj en `keyPressed` funktion der tillader at slå alarmen til og fra når der trykkes 's'
- *Time out:* Tilføj en timer, der sættes til 180 (3 sekunder) når alarmen går i gang, og som slår alarmen fra igen når den er nede på 0 (dvs. sætter alarmState til `"AKTIV"`).

Martin Dybdal & modified by DS
ds@di.ku.dk

Elbil-projektet - de sidste dele

Vi skal nu bruge en tilstandsmaskine til at få afsluttet elbilprojektet, men det kommer til at tage nogle skridt. Først skal vi have omskrevet logikken i bilen til at bruge en tilstandsmaskine i stedet for "carXVelocity". Vi har lige nu 3 tilstande:

- `"FORWARD"` - bilen kører mod højre
- `"BACKWARD"` - bilen kører mod venstre
- `"OUT_OF_FUEL"` - bilen er løbet tør for strøm og spillet er slut



- Når vi er i FORWARD tilstanden, flyttes bilen fremad, batteriet mister strøm
- Når vi er i BACKWARD tilstanden flyttes bilen bagud, batteriet mister strøm
- Når vi er i OUT_OF_FUEL tilstanden skal der ikke ske noget. Spillet er slut.

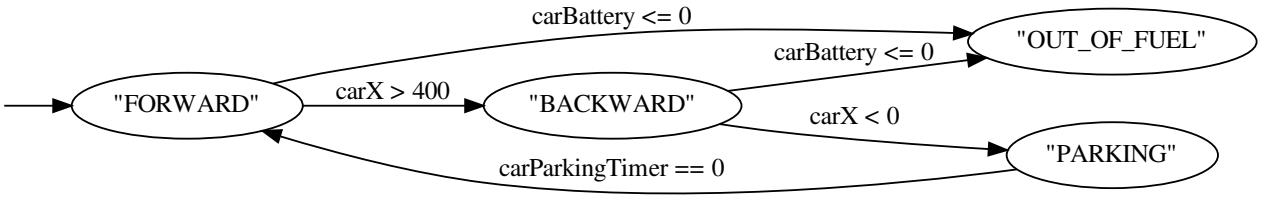
Koden skal have følgende struktur. Din opgave er at udfylde "..." med koden der flytter bilen, skifter tilstand og bruger strøm fra batteriet.

```
if carState == "FORWARD":
    ...
    if carX > 400:
        ...
    if carBattery <= 0:
        ...
elif carState == "BACKWARD":
    ...
    if carX < 0:
        ...
    if carBattery <= 0:
        ...
elif carState == "OUT_OF_FUEL":
    pass # pass means "do nothing"
```


Martin Dybdal & modified by DS
ds@di.ku.dk

Udvidet tilstandsmaskine

Vi skal nu have gjort så bilen parkerer hver gang den når venstre side af skærmen. Derfor udvider vi vores tilstandsmaskine med en parkeringstilstand:

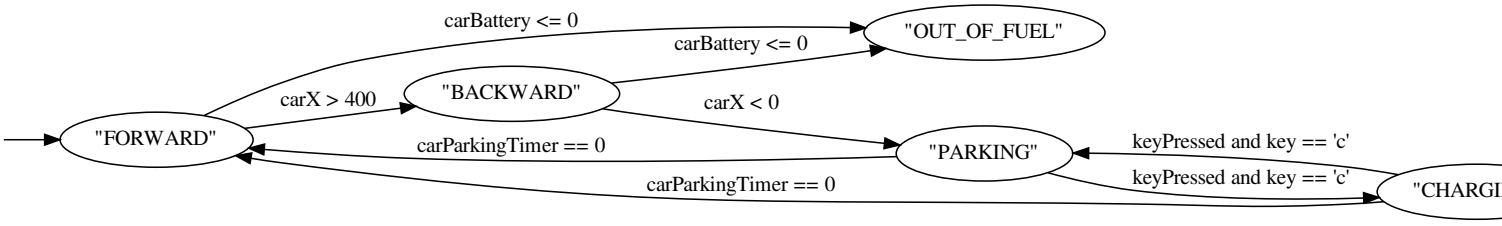


Samtidig skal der ske noget når vi skifter tilstand:

- Når vi skifter tilstand fra BACKWARD til PARKING, skal vi sætte en timer. Det gør vi ved at sætte en global variabel carParkingTimer til 600 (10 sekunder).
- Når vi er i PARKING tilstanden trækker vi en fra carParkingTimer i hver iteration af draw.

Kun opladning når der er parkeret

Den sidste udvidelse af tilstandsmaskinen gør at vi kun kan oplade bilen imens vi er parkeret.



Der tilføjes tre skift mellem tilstande:

- Tilstanden skifter frem og tilbage mellem PARKING og CHARGING når der trykkes på tasten 'c'
- Når man er i CHARGING tilstanden, skal bilen også begynde at køre igen når `carParkingTimer` er 0.

Samtidig skal der tilføjes kode så bilen lader automatisk så længe `carState == CHARGING`.

CO2 udledning

Den sidste del af projektet er at få CO2-udledningen til at afhænge af vindhastigheden. Vi opretter en global variabel `totalEmission` og sætter den til:

```
if carState == "CHARGING":
   totalEmission = totalEmission + 4 - windSpeed
```

Når windSpeed er 4 m/s vil der derfor ikke forurenes, når windSpeed er 3 vil derforurenes en lille smule og så videre.