# Pacman-spil

Hvad skal vi lave? Pacman er svaret. Når vi taler om at lave spil, er de ideer man kommer på ofte dem som udvider spillet på en interessant måde, men før vi kan gå til alt det interessante, skal vi lige få det basale til at køre først.

# Step 1: Pygame filen

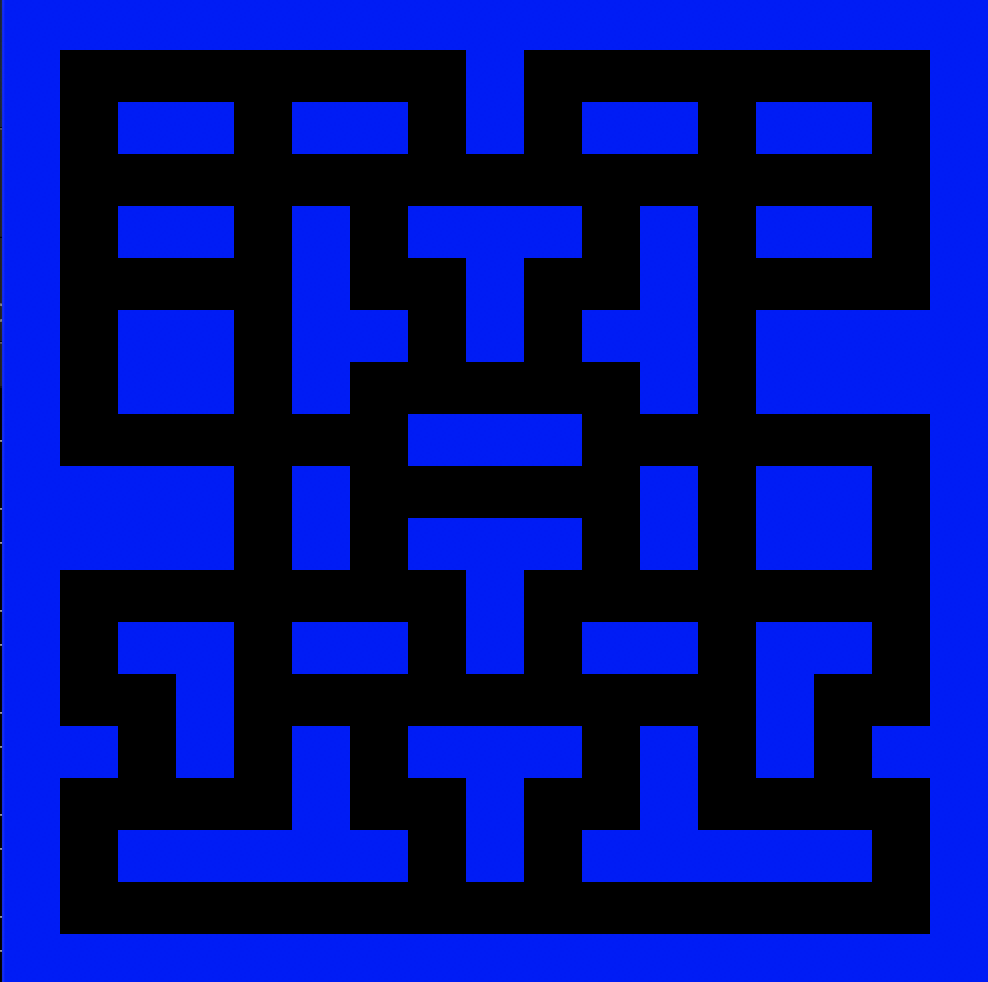
Ja før vi overhovedet kan få lavet noget som helst der ligner pacman, skal vi først lige få lavet en basal pygame fil som vi har gjort så mange gange før. Tast koden nedenunder, tjek om den virker (altså at der kommer en tom skærm, man kan lukke på krydset), hvis den virker så hop videre.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

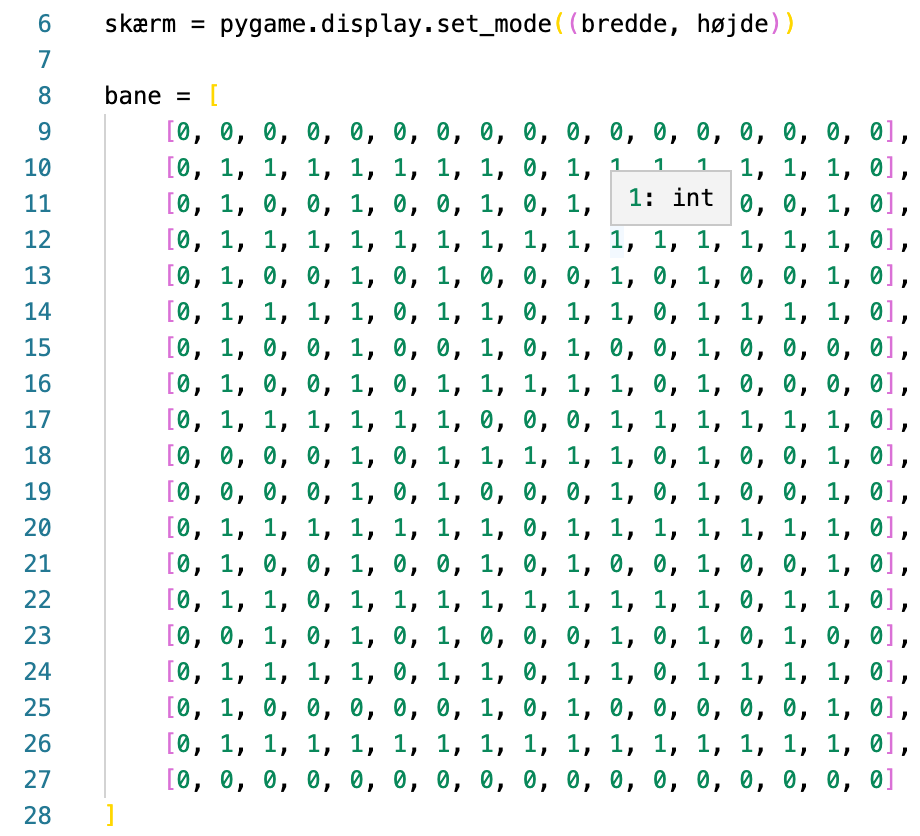
# Step 2: Banen

Banen i pacman er ret ikonisk, men hvordan kan vi repræsentere den i vores program, hvor vi både nemt kan tegne den, sørge for at pacman og spøgelserne kan bevæge sig rundt i den og lave flere forskellige baner?



Ja, svaret er (som det næsten altid er) lister. Vi kan repræsentere banen som en liste af lister, hvor den første liste indeholder alle rækkerne af felter og de rækkerne indeholder felterne. På den måde vil pacmans position bare være to indeks. Indtil videre vil vi bare have noget der er en sti og noget der er en væg, men dette kan muligvis blive udvidet i fremtiden, derfor bruger vi tal til at vise de individuelle blokke. Vi beslutter derfor at en væg er 0 og at en sti er 1. Men som i nok kan gætte er der en forfærdelig masse tal der skal indtastes, derfor har jeg været sød og uploadet det her (<http://korturl.dk/4win>)

Det indsættes i koden således (indrykningen betyder ikke noget):



Nu har vi repræsenteret vores bane i data. Nu vil vi også gerne kunne tegne den.

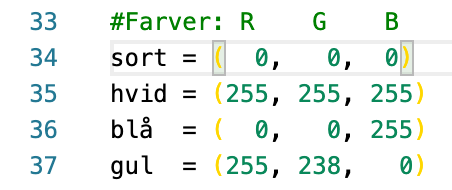
Da hver af tallene i listen ovenover repræsentere en blok og hver af listerne af en række, skal vi bare ”løbe” gennem hele banen for at vi har kommet gennem alle felter. Men for at vi tegner dem så at hele skærmen bliver fyldt ud af vores bane, skal vi have lavet nogle variabler der indeholder hvor stor hver blok skal være.

De defineres således:

Et billede, der indeholder objekt

Automatisk genereret beskrivelse

For at kunne tegne noget i pygame skal vi også kende farven af den ting vi tegner. Væggen skal være blå (eller hvad i lige kan finde på) og stien sort, men vi kan samtidigt definere nogle af de andre farver vi kommer til at bruge (bl.a. gul til pacman). Det gør vi sådan her:

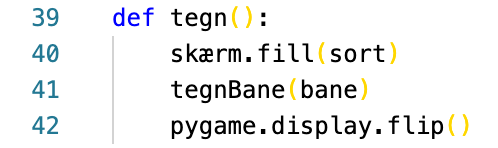


Nu har vi alt hvad vi skal bruge til at kunne tegne vores bane. Vi husker igen at vi løber igennem alle felterne (altså rækker og blokke i rækken), hvor vi så tegner blokke (blå hvis det er en væg, sort hvis det er en sti). Så herunder er funktionen:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu kan vi tegne vores bræt, til hukommelsen. Derudover skal vi også have det på skærmen. Men da vi kommer til at skulle tegne mere end banen bør vi lave en funktion der samler alle de ting vi skal tegne. Det gør vi i funktionen tegn, som vi definerer over tegnBane.



Så skal vi bare sørge for at kalde tegn i vores gameloop, så det ser sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

# Step 3: Pacman (vi starter med at tegne ham)

Sig til når i når herned.

Nu skal vi til at få pacman ind i spillet. For at gøre det nemmere for os selv vil vi bruge noget der hedder klasser. Klasser er basalt set bare en måde at samle data og funktioner sammen (nogle vil måske kunne huske space invaders). Når man har defineret en klasse kan man så lave ”instanser”, hvilket bare betyder kopier af det med ens specifikke værdier man kan bruge i resten af sit program.

Data delen af en klasse skrives i en funktion indenunder klassen, der hedder def \_\_init\_\_(self,…): Vi ser her ordet self, hvilket repræsentere værdierne der tilhøre klassen selv (husk at disse kan variere ved hver instans).

De funktioner der skal bruge den data der høre til klassen kalder man metoder, hvilket skrives som def … (self, ..): hvor … er et hvilket som helst navn på metoden og .. bare er argumenter.

Nu hvor vi kun vil tegne vores pacman skal vi tænke over hvilken data der er nødvendig. Da vi indtil videre bare tegner ham som en cirkel og vi ved at pygame kræver en position (x- og y-koordinat) og en radius, disse data vil vi derfor også have i vores pacman klasse. Denne position (altså de specifikke pixelkoordinater ift. hvor pacman skal tegnes på skærmen) afhænger jo af pacmans position på vores bane. Derfor vil vi også vide hvor han er på banen (altså de to indekser).

Efter farverne skriver vi datadelen af pacman klassen sådan her.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Læg mærke til hvordan at vi bruger self, både i argumenterne (til at sige at denne funktion har adgang til klassens data) og hvilke data der hører til klassen pacman.

Nu kan vi så også lave den funktion der rent faktisk tegner pacman. Da vi allerede har defineret klassens data skal vi også sørge for at bruge den til at tegne ham. Vi definerer derfor en metode der hedder tegn, der beregner positionen på skærmen og så tegner ham der. Vores pacman klasse ender med at se sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu har vi lavet alt i klassen, der skal bruges for at kunne tegne pacman så nu mangler vi bare at få lavet en instans af denne klasse. Nedenunder laver vi en instans af Pacman klassen (hvor vi placerer ham i 4, 4 på banen) og propper det i en variabel kaldet pacman (med småt!).



Så skal vi bare opdatere vores tegn funktion så vi sørger for at tegne pacman. Man kalder en metode ved bare at skrive et punktum og så navnet på metoden og parenteser (muligvis med argumenter i), så vi gør det samme med vores Pacman instans der ligger i pacman variablen.



Tjek om det virker.

# Step 4: Pacman nu med bevægelse

For at få pacman til at bevæge sig skal vi give ham en retning. Da man enten kan bevæge sig horisontalt eller vertikalt, hvor pacman skal kunne bevæge sig både frem og tilbage / op og ned. Vi giver derfor pacman to nye variabler, en for farten i x-retningen og en for farten i y-retningen henholdsvis xf og yf.

Et billede, der indeholder skærmbillede, tekst

Automatisk genereret beskrivelse



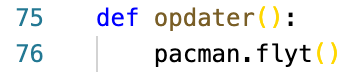
Nu skal vi så lave den funktion der rent faktisk skal flytte ham. Måde vi flytter ham på er ved at ændre på hans række og blok, altså hans indeks i banen. Men det er ikke bare så let fordi pacman skal jo ikke kunne bevæge sig ind over en væg, derfor skal vi sørge for at der hvor vi prøver at rykke ham hen ikke er en væg.

Vi ender derfor med følgende metode til pacman klassen (den skal skrives under tegn metoden, og skal indrykkes som på billedet nedenunder):

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu skal vi sørge for at flytte ham hver gang spillet opdateres. Da vi får flere ting vi skal flytte rundt på og opdatere giver det god mening at lave en funktion der tager sig af alle disse, ligesom med den generelle tegn funktion. Derfor laver vi følgende opdater funktion under tegn funktionen.



Nu skal vi bare kalde opdaterfunktionen i gameloopet, vi gør det lige under hvor vi kalder tegn funktionen. Prøv nu spillet. Hvad sker der? Bevæger han sig.

# Step 4.1: Pacman nu med bevægelse som brugeren kan lave om på

Grunden til pacman ikke bevægede sig tidligere var fordi vi ikke gav ham en retning at bevæge sig i. Men vi vil gerne have at spilleren kan styre hvilken vej pacman bevæger sig. Derfor skal vi til at have gang i vores input. For brugeroplevelsens skyld har jeg valgt at håndtere inputene til pacman lidt anderledes end normalt, hvis vi vil høre mere om dette så spørg (der er næsten altid en god grund, bag de skøre ting jeg gør).

Vi vil gerne have at pacman skal kunne bevæge sig til højre, venstre, op og ned. Derfor er det nok en god ide at bruge piltasterne. Dog skal vi også sørge for at der hvor vi vil have pacman hen ikke er en væg. Når vi har tjekket for dette kan vi så ændre på farten. Vi husker at vi har en x- og en y-fart. Hvis x-farten er negativ vil han bevæge sig til venstre fordi at indekset i en liste bliver mindre når man nærmer sig det første element. Det samme for y-farten hvis den er negativ vil han gå op. Så kan i næsten gætte hvad der skal til for at man går den modsatte vej.

Nå rigeligt snak, vores gameloop ender med at se sådan her ud (ps. Husk bare at lave om på jeres eksisterende loop):

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Prøv at se nu. Nu skulle i gerne kunne styre pacman.

# Step 5: Tag det roligt pacman

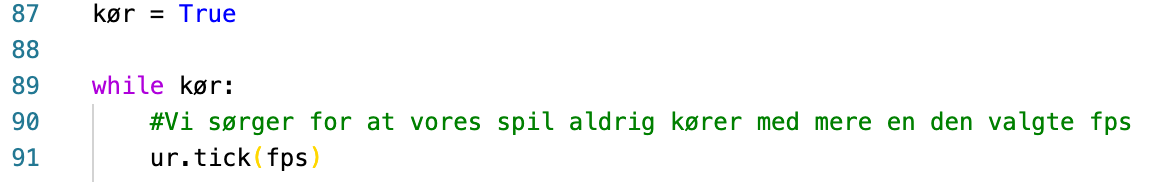
Da i prøvede jeres kode var der så noget usædvanligt ved måden pacman bevægede sig på? Gik det måske lidt stærkt, for det syntes jeg. I stedet for at pacman bare løber så hurtigt han kan så lad os prøve at sætte en grænse for hvor hurtigt det kan gå.

I kender måske allerede udtrykket fps, hvilket står for ”frames per. second” hvilket betyder billeder pr. sekund. Dette brugte vi i dino-spillet og nu skal vi bruge det igen. Vi kan nemlig sætte et max på fps altså hvor mange gange han skal flyttes i sekundet. For at vi kan det skal vi bruge to nye variabler nemlig et ur og en fps.

Et billede, der indeholder objekt

Automatisk genereret beskrivelse

Nu hvor vi har disse to skal bare sørge for at spillet holdes tilbage hvis det prøver at løbe for stærkt, det gør vi i gameloopet. I toppen af vores gameloop beder vi altså uret om at holde spillet tilbage hvis det køre for hurtigt toppen af gameloopet ender med at se således ud:

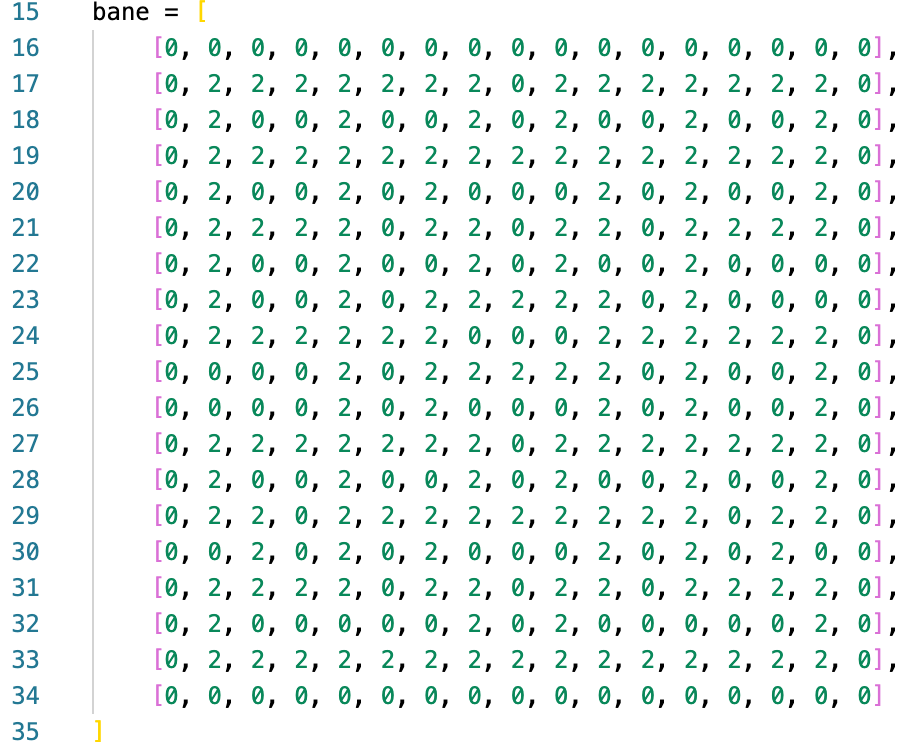


Hvor resten af gameloopet kommer efter helt uændret. Prøv det kører det for langsomt, for hurtigt? Prøv at ændre på fps variablen og se hvad der sker.

# Step 6: Pacman mangler mening

Pacman eksisterer nu i en verden, hvor han er fordømt til at løbe rundt i en labyrint der ingen ende har! Lad os prøve at give ham noget at lave. I pacman er en af de ting der gør det interessant at spille pointene. Der findes flere forskellige måder at få point på, men en af de mest ikoniske er de små gule prikker han skal samle op. Så lad os prøve at få lavet dem.

Disse gule prikker findes mange steder på banen, så lad os få lavet en blok der repræsenterer dem. I vores bane betyder 0 en væg og 1 en sti, så lad os gå lidt længere op af talrækken og sige at 2 skal stå for en gul prik. Vi starter derfor med at lave alle 1’erne om til 2’ere i vores bane:



I stedet for at gøre dette i hånden kan i lave et nyt dokument og søg-erstatte 1 med 2. (Det er noget nemmere).

Nu hvor vi har fået det på vores bane i dataen vil vi også gerne kunne se det. Vi skal derfor tage et kig på vores tegnbane funktion. Vi skal nu udvide den med endnu et elif-statement for hvis blokken er lig 2.

Så skal den nemlig tegne en sort firkant med en mindre gul cirkel. Nedenunder laver vi det ekstra elif-statement og funktionen der tegner den nye blok.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Prøv nu at kør koden og så skulle i gerne få følgende bane:

Et billede, der indeholder ur, tekst

Automatisk genereret beskrivelse

Hvis i får denne bane frem så kan i roligt går videre.

# Step 6.1: Pacman mangler mening (for han kan ikke bevirke den der er omkring ham)

I så nok at der ikke skete noget hvis vi bevægede pacman rundt, selvom han gik over pointene fik han ingen. Det må vi lave om på for den eneste grund til at vi lavede de her point var for pacmans skyld. Derfor vil vi nu sørge for at han kan få nogle point. Det hele stater med at vi skal holde styr på hvor mange han har fået indtil videre derfor laver vi endnu en variabel i pacman klassen. Så \_\_init\_\_ metoden ender med at se således ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu mangler vi bare at sørge for at han kan opfatte og dermed ændre på antal point når han går over et felt med et point i. Men vi skal også sørge for at han laver denne pointblok om til en normal blok efter vi har spist den. Dette gør vi alt sammen i pacmans flyt metode, der ender med at se således ud:

Et billede, der indeholder skærmbillede

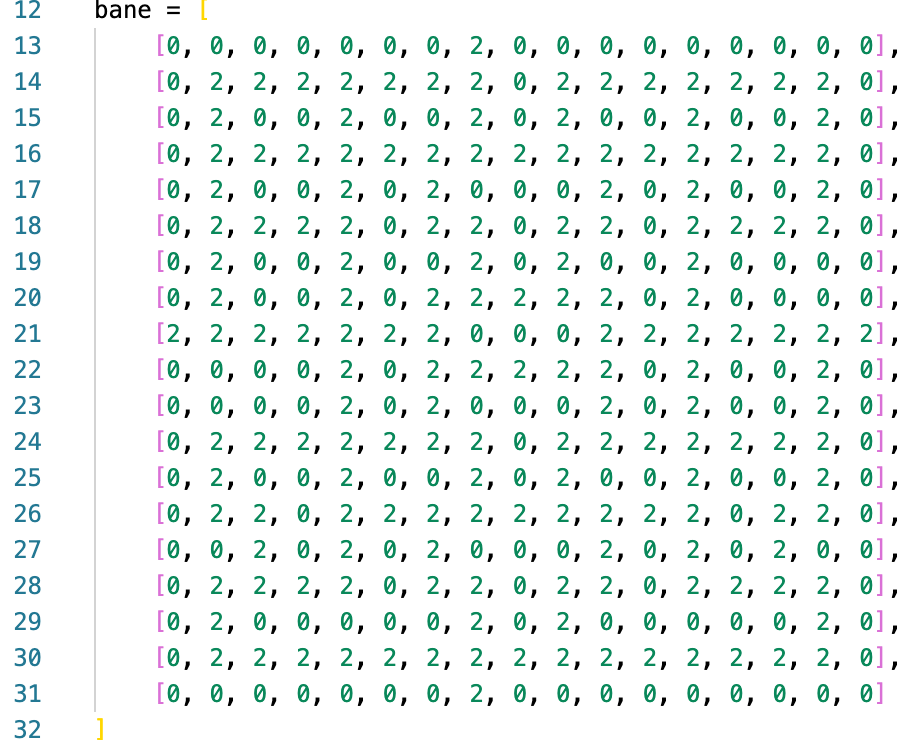
Automatisk genereret beskrivelse

Vi også gerne kunne se hvor mange point pacman har så sørg for at printe pacmans point i gameloopet. Det gør man ved følgende linje kode: print(pacman.point)

# Step 7: Pacman skal også kunne GÅ ”UPSIDE DOWN … AND ROUND AND ROUND”

I det rigtige pacman spil kan man også gå fra side til side, men lad os også sørge for at han også kan gå oppe fra og ned (og omvendt) bare for en sikkerhedsskyld.

Det første trin for at dette kan lykkes er at vi har en bane der understøtter sådanne bevægelser, så vi skal have lavet en passage der går fra den ene væg til den anden og oppe fra og ned. Så jeres bane kan ende med at se sådan her ud:



Prøv at se hvad der sker når i prøver at gå fra den ene til den anden side og op og ned. Mærkeligt ikke?

# Step 7.1: Pacman skal også kunne GÅ ”UPSIDE DOWN … AND ROUND AND ROUND” og stadig kunne ses og ikke CRASHE!!

Ja, der sker nogle mærkelige ting det må man sige. Så vi bliver nødt til at gøre noget når vi nærmer os de kanter. Vi kan derfor kun rykke pacman som vi gjorde tidligere når vi ikke er på den yderste blok. Derefter er der to hoved tilfælde. Et for den horisontale bevægelse og et for den vertikale bevægelse. Derfra skal vi håndtere de to under tilfælde der er for den horisontale er det om den går fra venstre til højre eller omvendt. For det der skal ske er at blok/række indekset skal ”nulstilles”. Men nu vil jeg lade koden tale. Læg mærke til if-statementet der er tilføjet før den oprindelige flyt, derudover er det kun de to elif-statements der skal tilføjes til pacmans flyt metode.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

# Step 8: Spøgelser

Nu hvor pacman kan bevæge sig og han har en grund til at bevæge sig (pointene) skal vi også sørge for at der er noget sjovt og udfordrende til at få pacman til at bevæge sig. I pacman spillet er det spøgelserne, men hvordan fungerer de egentligt? Godt spørgsmål og ved du hvad det forklarer denne video(<http://kortlink.dk/24vpw>) utrolig godt så se lige ca. 10 minutter ind for at få en bedre forståelse af hvad vi kommer til at gøre i den følgende kode

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseI den video blev der givet en lang række beskrivelser af hvordan alle facetterne af spøgelserne i pacman fungerer. Vi vil starte med bare at kunne få et spøgelse til at bevæge sig hen mod et mål som det bliver beskrevet fra 2:00 til 4:00. Før vi kan gøre skal vi først få lavet spøgelserne, det gør vi ved at lave en spøgelses klasse der virkelig meget ligner pacmans. I første omgang lader vi flyt metoden stå tom. Læg mærke til at vi når vi laver spøgelset skal give det en farve, hvilket vi ikke gør i pacman klassen, da der kun skal være en instans af pacman, men flere af spøgelserne. Vi giver derudover også vores et spøgelse et mål, som det skal bevæge sig henimod, for at være sikre på at vi gør det rigtigt tegner vi her i udviklingsprocessen også målet som et lille rektangel. Spøgelses klassen kommer i første omgang til at se således ud og vi skal skrive den lige under pacman klassen.

Nu kan vi efter vores pacman variabel lave det første spøgelse hvilket hedder blinky hvilket vi derfor også kalder variablen for, derefter sørger vi for at tegne og flytte blinky (selvom at flytte koden ikke gør noget endnu), hvilket vi gør ved at lave om på vores tegn og opdater funktioner som vist i koden nedenunder.

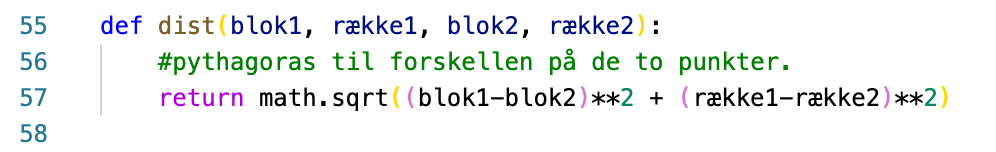
Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

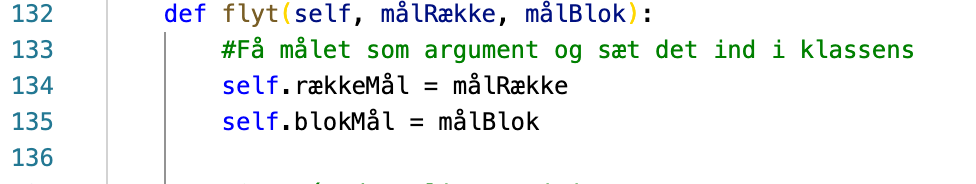
Prøv at se hvad det gør, ps. Spøgelset skal ikke bevæge sig endnu.

# Step 8.1: Få spøgelset til at bevæge sig

Nu skal vi så få spøgelset til at bevæge sig, hvilket vi gør (som forklaret i videoen) ved først at få målet for spøgelset, hvilket vi får ind som et argument, derefter finder vi så de mulige steder vi kan bevæge os hen. Det er muligt at bevæge sig et sted hen som ikke er en væg og ikke er der man var før. For at finde ud af hvilken af disse man skal vælge, kigger man på hvor langt der er i fugleflugt fra spøgelset til målet, hvor den mulige position der er tættest, vil være den der bliver valgt. Denne måde man afgør om hvor langt der er fra en position til en anden er en meget generel funktion som vi også gerne vil kunne bruge udenfor spøgelsesklassen, derfor skal vi lave en funktion der netop gør dette udenfor spøgelsesklassen. Vi laver derfor en funktion kaldet dist lige før pacman klassen, koden til denne funktion ser således ud.



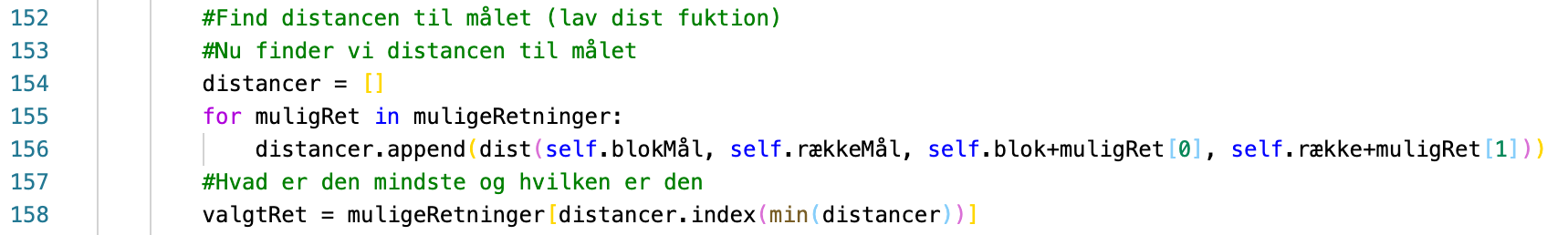
Nu kan vi så begynde at skrive koden til spøgelsets flyt metode som beskrevet i tekststykket ovenover, jeg vil i nedenstående stykke beskrive hver del af metoden, hvor der til sidst vil være et overblik over hele metoden.



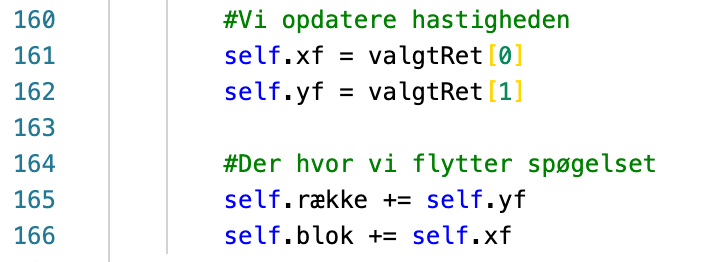
Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseI den første del af metoden tager vi bare målrækken og målblokken og sætter spøgelsets egen tilsvarende variabler til de givne argumenters værdi.

Derefter begynder vi at finde ud af hvilke retninger der overhovedet er muligt at bevæge sig i, læg her mærke til at retningerne er det samme som hastighed. Vi starter med at lave en tom liste til de mulige retninger, derefter løber vi igennem alle retningerne, hvor vi først tjekker at spøgelset ikke går ud fra banen, derefter at det ikke er en væg og at det ikke går tilbage, hvis alle disse betingelser er opfyldt sætter vi den mulige retning i muligeRetninger listen.



Nu skal vi så finde ud af hvilke af disse retninger er den bedste. Det gør vi ved at lave en liste der indeholder distancerne fra de muligeretninger til målet, hvor vi derefter vælger den mindste og finder det tilsvarende indeks for at finde den valgte retning.



Efter vi har fundet den valgte retning skal vi bare sætte spøgelsets hastighed til den og flytte spøgelset.

Nedenunder er den samlede kode tjek lige at alt ser nogenlunde ens ud og så prøv at kør det, følger spøgelset efter dig?

# Et billede, der indeholder skærmbillede Automatisk genereret beskrivelseStep 9: Få spøgelset til at bevæge sig lidt langsommere

Spøgelset bevæger sig godt nok hurtigt hva! Ja, det er lidt for svært at slippe væk fra dem så den fart må vi lige få lavet lidt om på, det kan vi heldigvis også fordi vi er mega kodegenier. Okay så for at få spøgelset til at bevæge sig lidt langsommere gør vi bare så at spøgelset ikke bevæger sig hver gang pacman får lov til at bevæge sig, men kun en gang imellem. Vi kan derfor lave en bevægSpøgelse variabel, der tæller op hvor mange gange det har været siden spøgelset bevægede sig og efter et eller andet antal gange (jeg har valgt 4) bevæger vi det og starter forfra. Vi laver bevægSpøgelse variablen lige efter blinky variablen og så laver vi bare følgende om på opdater funktionen, som vist i koden nedenunder:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Virker det bevæger spøgelset sig langsommere?

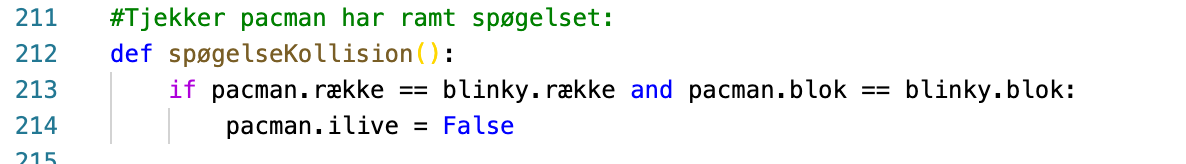
# Step 10: Spøgelser er dødelige

Ja, det er sandt spøgelser er altså dødelige, det er de i hvert fald i pacman, så det skal vi også få dem til at være. Så lad os starte med at tilføje en variabel til pacman klassen der kan fortælle om pacman er i live eller ej, vi kalder denne variabel ilive. Herunder er den nye \_\_init\_\_ metoden i pacman:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

For at bedømme om pacman skal dø skal vi tjekke om han er ramt ind i spøgelset eller ej, vi laver derfor en funktion vi kalder spøgelseKollision, vi sætter den under tegnBlokPoint funktionen, den ser sådan her ud:

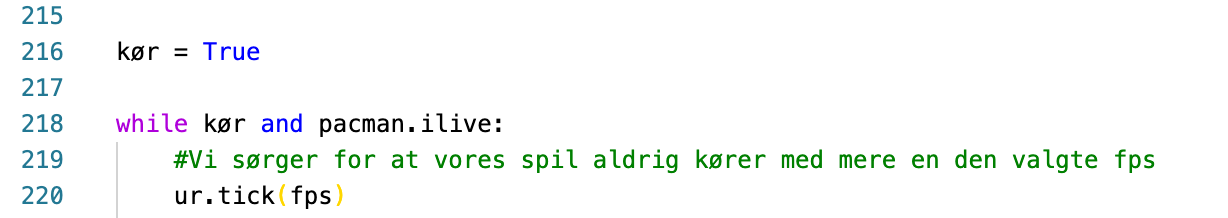


Vi kalder nu denne funktion i vores opdater funktion, som ender med at se sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu skal vi så stoppe spillet hvis pacman ikke er i live. Derfor laver vi om på gameloopet så det ikke kun afhænger af kør variablen, koden der ender derfor med at se sådan her ud:



Dør pacman?

# Step 11: Pinky

Vi vil nu til at lave flere spøgelser, som i nok kan huske fra videoen har hver af spøgelserne hver deres unikke måde at bevæge sig på, vi har allerede lavet den første, nemlig blinky, der bare bruger pacmans koordinater som mål. Nu skal vi til at lave det næste spøgelse nemlig pinky, der bruger pacmans koordinater, men bare fire foran ham. Dog skal vi lige sørge for at vores resterende kode kan understøtte flere spøgelser, så det starter vi med.

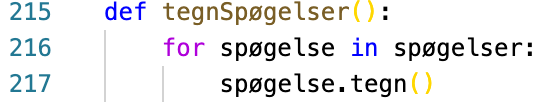
Det første vi skal gøre er at lave vores nye spøgelse det gør vi bare ved at lave en ny variabel kaldet pinky, vi giver den et start koordinat og farven pink (som vi skal definere ved de andre farver rgb værdien er: (252,134,170)), dette nye spøgelse laver vi lige under blinky.



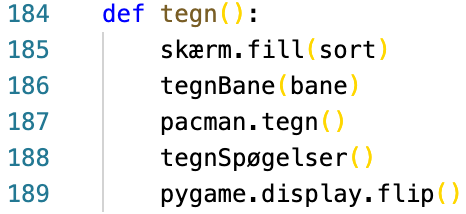
For at understøtte flere spøgelser laver vi en liste af alle spøgelserne lige nedenunder pinky.



Nu skal vi så bruge denne liste i funktioner, hvor vi bare kalder en af spøgelsernes metoder eller bruger noget af deres data som f.eks. når vi tegner spøgelserne eller vi tjekker om de har ramt pacman. Vi laver derfor en funktion vi kalder tegnSpøgelser som ser således ud vi sætte den over spøgelseKollision:



I stedet for blinky.tegn() i vores tegn funktion skal vi nu erstatte den med tegnSpøgelser() så vores tegn funktion ser således ud:

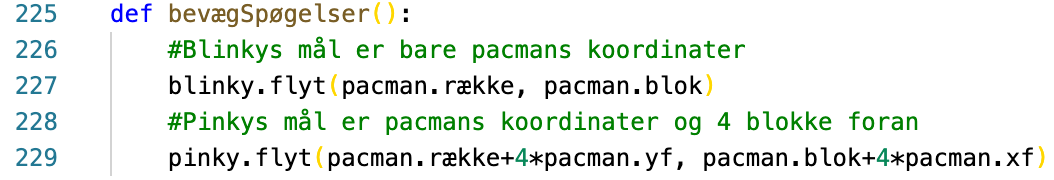


Nu kan vi gøre det samme bare for spøgelseKollision funktionen, hvor vi erstatter blinky med spøgelse, så den ser sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

I takt med at vi får flere spøgelser er det en god ide at få delt koden op i mindre, mere forståelige dele. Derfor skal vi også lave en separat funktion der flytter alle spøgelserne. Denne funktion kommer dog til at skulle håndtere en separat situation for hver af spøgelserne, så derfor kan vi ikke bare bruge spøgelser listen. Vi kalder denne funktion bevægSpøgelser, vi sætte den under spøgelseKollision og den kommer til at se således ud:



Nu skal vi bare sørge for at få kaldet denne funktion i vores opdater funktion i stedet for blinky.flyt, dog vil vi stadig gerne beholde at spøgelserne kun bevæger sig en gang imellem, vores opdater funktion ender dermed med at se således ud:

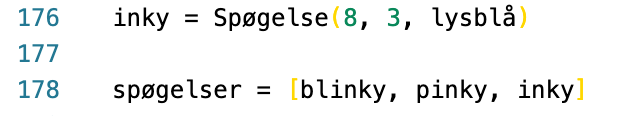
Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

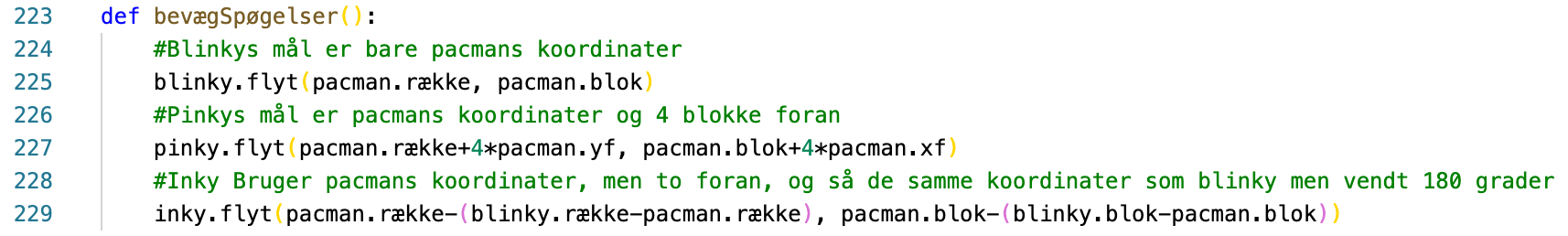
Virker det som forventet? Er der to spøgelser, hvor er pinkys mål?

# Step 12: Inky

Da vi i det tidligere trin har lavet alt grundarbejdet for at kunne tilføje nye spøgelser bliver det utroligt nemt at tilføje de næste. Det eneste vi skal gøre er at lave variablen, tilføje dem til spøgelser listen og kalde deres flyt funktion i bevægSpøgelser med det mål spøgelset nu har. Det gør vi så her, vi laver inky (hvor vi giver den en lysblå farve vi skal lave oppe ved de andre farver, rgb værdien er (27,177,230) ) således og tilføjer ham i spøgelser listen:



Derefter skal vi bare skrive koden til inkys flyt metode i bevægSpøgelser. Før vi kan det skal vi lige huske hvordan det var vi fandt hans mål. Inkys mål fandt vi ved at tage den blok to foran pacman, hvor man derfra fandt distancen til blinky, hvor at den omvendte (altså den vendt 180 grader) vil være inkys mål. Det kan alt sammen skrives noget mere simpelt som ses nedenfor, hvilket er hvordan bevægSpøgelser funktionen skal se ud:



Tjek om der er tre spøgelser og hvor inkys mål er.

# Step 13: Clyde

Nu er vi kommet til det mest besværlige spøgelse, nemlig clyde. Clyde har nemlig pacmans koordinater som mål, så længe han er 8 (hvor vi siger 4 fordi vores blokke er større end det originale spil) eller flere blokke væk fra pacman ellers er hans mål det nederste venstre hjørne, men lad os ikke tage sorgerne på forskud. Vi skal lave clyde på næsten præcis samme måde som de to forrige spøgelser altså først ved at lave hans variabel, hvor han får farven orange (rgb: (249,126,22)) som defineres oppe ved de andre farver, og derefter ved at sætte denne variabel ind i spøgelser listen:

Et billede, der indeholder objekt

Automatisk genereret beskrivelse

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseNu skal vi så til at kigge på clydes bevægelse. Når det kommer til stykket er det rent faktisk ikke så slemt, vi skal bare tjekker om der er mere end 8 blokke til pacman, altså om distancen imellem dem er over 8 (hvor vi siger 4, fordi vores blokke er større), hvis ja så kalder vi clydes flyt metode med pacmans koordinater ellers kalder vi dem med det venstre hjørne. Vores bevægSpøgelser funktion ender derfor med at se således ud:

Er der fire spøgelser nu? Hvad med clydes mål skifter det som vi har beskrevet?

# Step 14: Hvordan spiser man spøgelser?

Hvad er det i pacman der gør man kan spise spøgelserne? Det er de kæmpe store piller, så det er også det vi skal have ind i vores spil. Det gør vi ligesom vi fik de andre piller ind, i stedet for 2 kalder vi de store piller 3. Så ændrer nogle af dine 2’ere i din bane til 3. Nu skal vi sørge for at vi også kan se dem, derfor skal vi lave lidt om på tegnBane funktionen.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu hvor vi kalder en funktion tengPowerBlokPoint skal vi også sørge for at den funktion eksisterer. Den ligner meget tegnBlokPoint, men hvor cirklen har en større radius. tegnPowerBlokPoint funktionen ser derfor sådan her ud:

Et billede, der indeholder værktøj

Automatisk genereret beskrivelse

Prøv at kør det, kan du se dem, men hvorfor kan pacman ikke spise dem? Godt spørgsmål.

For at pacman skal kunne spise dem skal vi ind i hans flyt funktion, lige under der hvor vi styrer pointene. Det vi vil gøre er at vi skal skifte til en anden mode, nemlig flygt ”moden”, hvilket vi repræsenterer som 3. Derudover skal vi sørge for at sætte blokken til en sti efter pacman har spist power pillen. Så slutningen af pacmans flyt funktion ender med at se sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Hvad gør det, skifter spøgelserne mode?

# Step 15: Spis spøgelserne

Når der er blevet spist en power pille skifter spillet mode, den mode skal sørge for at spøgelserne opfører sig anderledes. For at vi kan det skal vi sørge for at vi bruger mode variablen hele vejen igennem vores program, første måde vi sørger for det er ved at lave variablen så den kan anvendes globalt. Vi definerer derfor en variabel mode og sætter den til nul, under hvor vi har defineret vores spøgelser…

Et billede, der indeholder skærmbillede

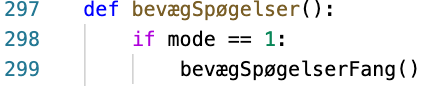
Automatisk genereret beskrivelse

Nu skal vi til at sørge for at spøgelserne rent faktisk bevæger sig efter de her modes. Så det skal vi gøre ved vores bevægSpøgelser funktion. Vi starter med at lave navnet på vores bevægSpøgelser funktion om til bevægSpøgelserFang, da den kode kun passer når vi er i mode 1. Så det ender med at se således ud.

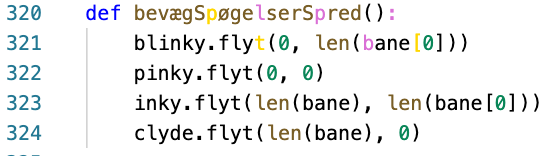
Et billede, der indeholder indendørs, skærmbillede

Automatisk genereret beskrivelse

Nu kan vi derover så lave en funktion bevægSpøgelser, der kan skifte mellem hvordan spøgelserne skal bevæge sig afhængigt af hvilken mode spillet er i. Den funktion starter ud med at se således ud:



Nu kan vi så skrive koden der kan flytte spøgelserne som de skal når de er i mode 2. Når de er i mode 2 skal de gå mode hvert sit hjørne, her kan vi bruge flyt funktionen for hver af spøgelserne, men hvor at målet bare er et af hjørnerne. Vi laver derfor en funktion bevægSpøgelserSpred som ender med at se sådan her ud:



Nu kan vi så tilføje endnu en mode til bevægSpøgelser, bevægSpøgelser kommer derfor til at se sådan her ud:



Nu har vi to af tre modes. Nu mangler vi bare mode 3 også kendt som flygt mode. Reglerne for denne mode er dog lidt anderledes og kræver vi laver en ny funktion (eller metode) i Spøgelses klassen vi kalder flygt. Den ligner utrolig meget flyt fordi vi også starter med at generere alle de mulige retninger, men i stedet for at tage den der har kortest til målet (hvilket der ikke engang er) vælger den bare en tilfældig. Når vi har noget med tilfældighed at gøre skal vi også huske at importere random modulet. Flygt funktionen i spøgelses klassen (under flyt) ender med at se sådan her ud:

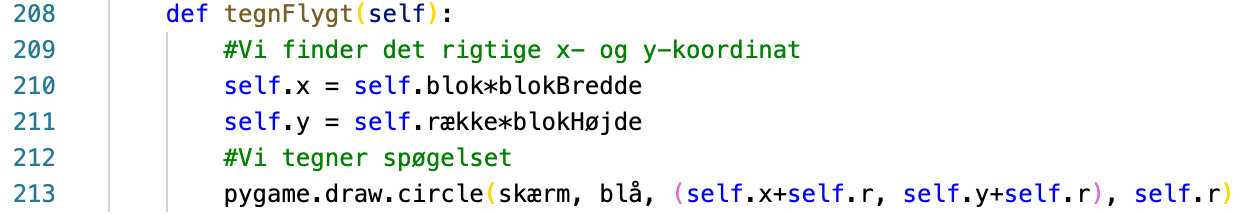
Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseNår vi har skrevet den ind i spøgelses klassen kan vi tilføje mode 3 til vores bevægSpøgelser funktion, da vi ikke skal angive nogle specifikke mål kan vi bare løbe igennem alle spøgelserne og flytte dem en efter en, bevægSpøgelser funktionen ender med at se således ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu skulle spøgelserne gerne bevæge sig forskelligt afhængig af hvilken mode de er i. Dog kan man ikke se at de skifter modes. Det gør heller ikke noget ift. mode 1 og 2, men mode 3 der vil vi gerne have at de har den traditionelle blå farve. Derfor skal vi få brugt modesne når vi tegner vores spøgelser også. Vi starter derfor med at lave en metode i vores spøgelses klasse kalde tegnFlygt, som vi sætter under tegn. Den fungerer på meget samme måde, den er bare blå i stedet for spøgelsets farve.



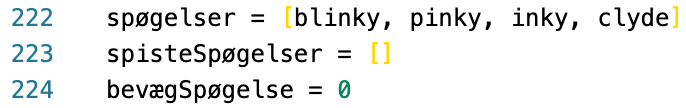
Nu skal vi så også sørge for at denne metode bliver brugt, det skal den hvor vi tegner vores spøgelser, nemlig i funktionen tegnSpøgelser. Vi gør sådan at hvis de ikke er i mode 3 skal de bare tegnes som sædvanlig ellers skal de bruge tegnFlygt. tegnSpøgelser funktionen ender med at se sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu skulle spøgelserne gerne blive blå når vi spiser power pillen. Men hvad sker der hvis vi prøver at spise dem? Nej, det er dem der skal dø ikke os!

Vi mangler stadig at sige til spillet at reglerne har ændret sig, så i stedet for at vi dør så skal pacman få point når vi spiser et spøgelse. Det kan vi gøre ved at introducere modesne i vores opdater funktion. Men før vi kan det skal vi lige have lavet den funktion der kan tjekke om vi rent faktisk har spist nogle spøgelser. Det der skal ske med spøgelser når vi har spist dem er at de skal fjernes fra spøgelses listen og proppes i en separat liste for spiste spøgelser, denne kalder vi spisteSpøgelser. Lav en variabel spisteSpøgelser ved spøgelser listen.



Nu kan vi så lave spøgelseSpis funktionen. Det den skal gøre er at spørge om pacman har ramt et spøgelse, hvis han har det skal pacman få 100 point og spøgelset skal fjernes fra spøgelser listen og proppes i den nye spisteSpøgelser liste. Det er nogle symboler i nok ikke har set før, spørg og så skal vi nok forklarer (der skulle nemlig lidt magi til at få det til at virke)

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Så skal vi bare få modesne og spøgelseSpis ind i vores opdater funktion. Så det gør vi da bare, opdater funktionen ender med at se således ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

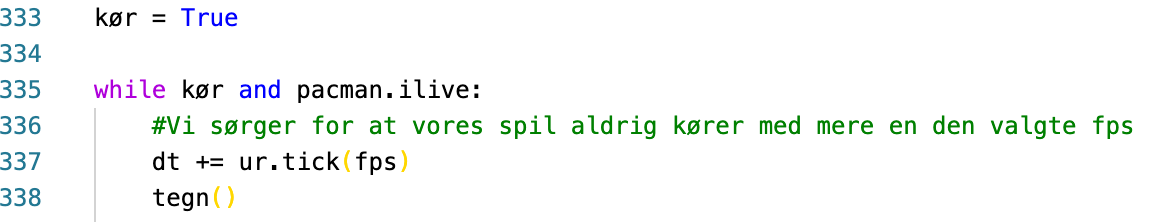
Nu skulle spøgelserne gerne skifte farve og forsvinde når man spiser dem. PRØV.

# Step 16: Kommer spøgelserne aldrig tilbage?

Der er som at når man først har spist en power pille kan man spise alle spøgelserne hele tiden og det er ikke sådan at det rigtige pacman virker. Nu skal vi sørge for at vi skifter imellem de forskellige modes. Som i nok kan huske fra videoen var der nogle ret specifikke tidstabeller, men dem vælger jeg ikke at følge, men hvis i har overskud til det skal i for guds skyld bare gøre det. Når tilbage til koden. For at vi kan skifte mellem modesne efter et specifikt stykke tid skal vi huske hvor meget tid der rent faktisk er gået. Vi laver en variabel der holder styr på hvor lang tid der er gået siden sidste gang vi skiftede mode, vi kalder den dt hvilket står for delta tid. Vi laver den der hvor vi også laver vores spøgelser:



Nu skal vi så få noget tid ind i denne variabel. Det gør vi ved at bruge noget af den kode der sørge for at spillet ikke kører for hurtigt, for den funktion (tick funktionen) returnerer faktisk hvor mange millisekunder der er gået siden sidste frame. Vi lægger derfor bare denne tid til vores dt. Starten af vores game loop ender derfor med at se sådan her ud:



Nu skal vi også bare sørge for at skifte mellem modesne. Vi laver derfor en funktion vi kalder skiftMode, der skal tjekke om der er gået lang nok tid til at skifte mode. Jeg har valgt at sige at der skal gå 7 sekunder (7000 millisekunder) før den må skifte mode. Hvor den så skal skifte til den modsatte mode, undtagen hvis den har været i mode tre hvor den skal skifte til mode 1. Koden til denne funktion ser sådan her ud:

Et billede, der indeholder tekst

Automatisk genereret beskrivelse

Denne funktion skal vi kalde til sidst i vores opdater funktion. Nu skifter den mellem modesne automatisk. Der er dog også et andet sted der bliver skiftet mellem modesne, det sker nemlig når pacman spiser en power pille, hvor vi så også skal huske at få sat dt tilbage til 0. Den sidste del af pacmans flyt funktion ender derfor med at se sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Nu skulle spøgelserne gerne skifte tilbage fra deres blå skikkelse.

# Step 17: Spøgelsets genopstanden?

Det er ikke så sjovt når der slet ikke er nogle spøgelser tilbage vel? Nu skal vi sørge for at spøgelserne i listen af spisteSpøgelser bliver taget af så de kan komme tilbage i kampen. For at spøgelserne liver op skal de tilbage foran deres hjem, jeg har sagt at det er ved blok (7,8) i kan jo vælge et andet mål. Men det vi skal er faktisk bare at flytte de spisteSpøgelser mod dette mål og hvis de rør det så skal de tilbage i spøgelser listen og ud af spisteSpøgelser listen. Koden til denne funktion ender med at se således ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Vi skal bare kalde denne funktion i opdater og så er disse spisteSpøgelser i bevægelse. Nu skal vi også bare lige kunne se disse skikkelser i deres færd mod hjemstedet så de ikke bare kommer ud af ingenting. Så vi skal tegne de klassiske øjne, det gør vi ved denne funktion:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Denne funktion kalder vi i tegn funktionen. Prøv så kommer spøgelserne tilbage og kan du se deres smukke øjne. Nu har vi faktisk alle de centrale dele af gameplayet! Nu er det tid til at få kigget lidt på det grafiske!

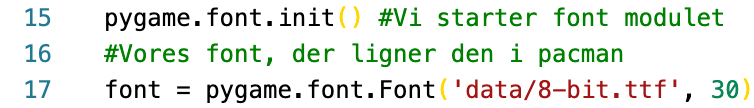
# TID TIL GRAFIK OG PUDSNING

# Step 18: Scoren på skærmen

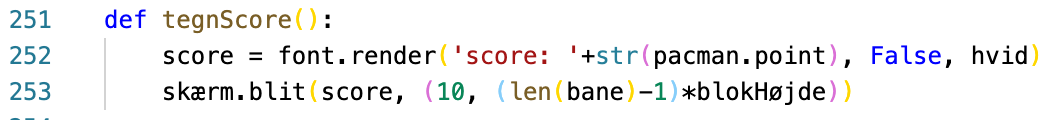
Hvem kigger i konsollen mens de spiller, svaret er ingen. Scoren skal stå på skærmen, det er helt essentielt. Men før vi kan begynde og bruge en masse flotte billeder, fonte og lyde i vores spil skal vi lige have disse filer. Dem i skal bruge kan i finde her: [www.kortlink.dk/25363](http://www.kortlink.dk/25363)

Denne folder skal i sætte ind i den folder jeres spils fil også er i. (spørg hvis i ikke lige forstår)

Vi vil starte med at få scoren på skærmen så spilleren ved hvor god han/hun er. Men i stedet for at bruge en traditionel font har jeg vedlagt en 8-bit font i den data mappe i har downloadet så den vil vi bruge. For at få denne font i vores spil skal vi først starte font modulet og så loade fonten ind (det gør vi lige under hvor vi har defineret skærm)



Nu kan vi så bruge denne font til at skrive scoren på skærmen som vi gør i en funktion tegnScore, der ser således ud:



Denne funktion kalder vi som den næstsidste i tegn. Så tegn ender med at se således ud:

Et billede, der indeholder skærmbillede, tekst

Automatisk genereret beskrivelse

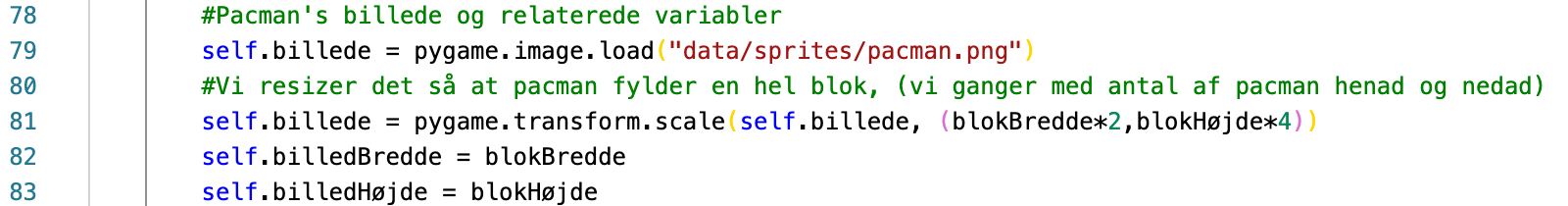
Kan i se scoren?

# Step 19.1: Pacmans billede

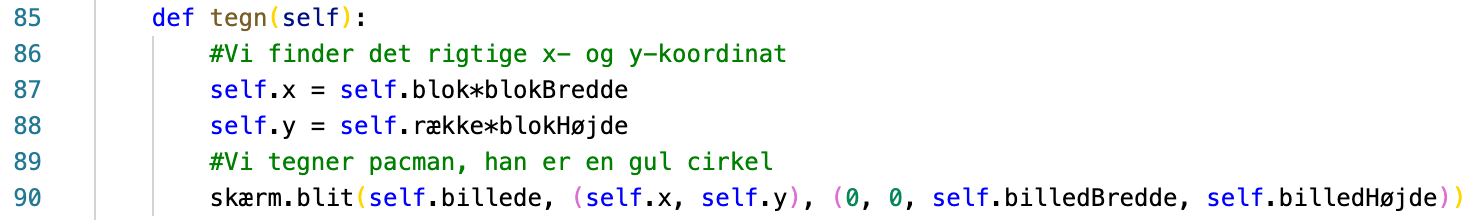
Pacman er ret ikonisk og den gule cirkel vi anvender lige nu ligner ham ikke helt. Prøv at tage et kig i mappen i downloadede i sidste trin. Kan se pacmans billede?



I ser ovenover hvordan pacmans billede ser ud. Man ser hurtigt at der er flere pacmen end vi har brug for, det skyldes at det ikke er helt traditionelt billede, men noget kaldet et spritesheet. Det betyder at vi kun viser en lille del af billedet af gangen, afhængigt af hvilken af pacman vi vil vise. Vi ser at der er fire rækker, der hver peger en forskellig retning og i hver række er der et billede med mere eller mindre åbnet mund. Så rækken vi bruger skal afhænge af hvilken retning pacman går og de to forskellige billede skal vi bare bruge til at skifte imellem så det ligner han rykker sig. Men lad os starte med bare at få billedet ind i vores program. Vi vil sætte dette billede ind i pacmans klasse, så vi laver en variabel kaldet billede. Vi skal dog sørge for at pacman fylder hele feltet ud så derfor justerer vi på hans størrelse så det har samme dimensioner som en blok og til sidst laver vi også to variabler, der siger hvor stort hvert billedes dimensioner er. Vi tilføjer derfor følgende kode til pacman klassens \_\_init\_\_ funktion:



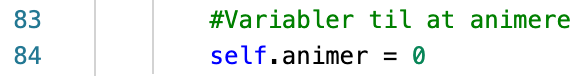
Nu har vi fået billedet ind, nu skal vi også bare sørge for at vise dette, vi laver derfor om på pacmans tegn funktion så den i stedet for at tegne en cirkel tegner billedet. Men nu skal vi huske at billedet indeholder mange billeder, derfor skal vi specificere hvor fra i billedet vi vil vise fra og hvor meget vi vil vise derfra. Det gør vi med pygames blit funktion der tager argumenterne skærm.blit(billede, (xSkærm, ySkærm), (xPåBillede, yPåBillede, breddeAfUdklip, højdeAfUdklip)). Vi starter med bare at bruge det første billede så vi tegner fra 0, 0 på billedet og så bare en billedbredde og billedhøjde hen. Tegn funktionen ender derfor med at se sådan her ud.



Nu skulle der gerne være et billede af pacman.

# Step 19.2: Pacman skal tygge

Nu har vi fået pacman på skærmen, lad os nu få ham til at tygge. Det gør vi ved at skifte mellem de to billeder i rækken. Da vi kun har to billeder vi skal skifte mellem, kan vi skifte mellem 1 og 0 ved at bruge not funktionen. For at holde styr på hvilket billede vi er på i rækken, laver vi en variabel der skal op i pacman klassens \_\_init\_\_ funktion, kaldet animer:



Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseNu skal vi bare sørge for at skifte mellem billederne hver gang vi tegner pacman. Det gør vi ved at ændre animer og ved at forskyde hvor vi tegner fra afhængigt af animer variablen. Så pacmans tegn funktion ender med at se således ud:

Nu skulle pacman gerne se ud som om han tygger.

# Step 19.3: Pacman skal pege i den rigtige retning

Nu mangler vi bare at sørge for at pacman vender i den rigtige retning. Det kan vi sørge for ved at bestemme hvilken række han skal tegnes ud fra afhængigt af hvilken retning han bevæger sig i. Vi starter med at lave variablen der skal indeholde hvilken række vi vil tegne vores pacman ud fra. Denne kalder vi billedRække og sætter den under animer variablen vi definerede i forrige opgave:

Et billede, der indeholder objekt

Automatisk genereret beskrivelse

Sådan nu skal vi så bare ændrer denne ift. hvilken retning pacman vender. Vi ser på billedet og ser at rækkerne svarer til følgende retninger:

Række 0 : Venstre, Række 1 : Højre, Række 2 : Op og Række 3 : Ned.

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseVi vælger derfor at tjekke hvilken retning vi bevæger os i i tegn funktion og sætter billedRække variablen til det rigtige. Derudover skal vi også sørge for at vi ændrer hvor vi viser billedet fra så vi bruger billedRække variablen. Derfor ender pacmans tegn funktion med at se således ud:

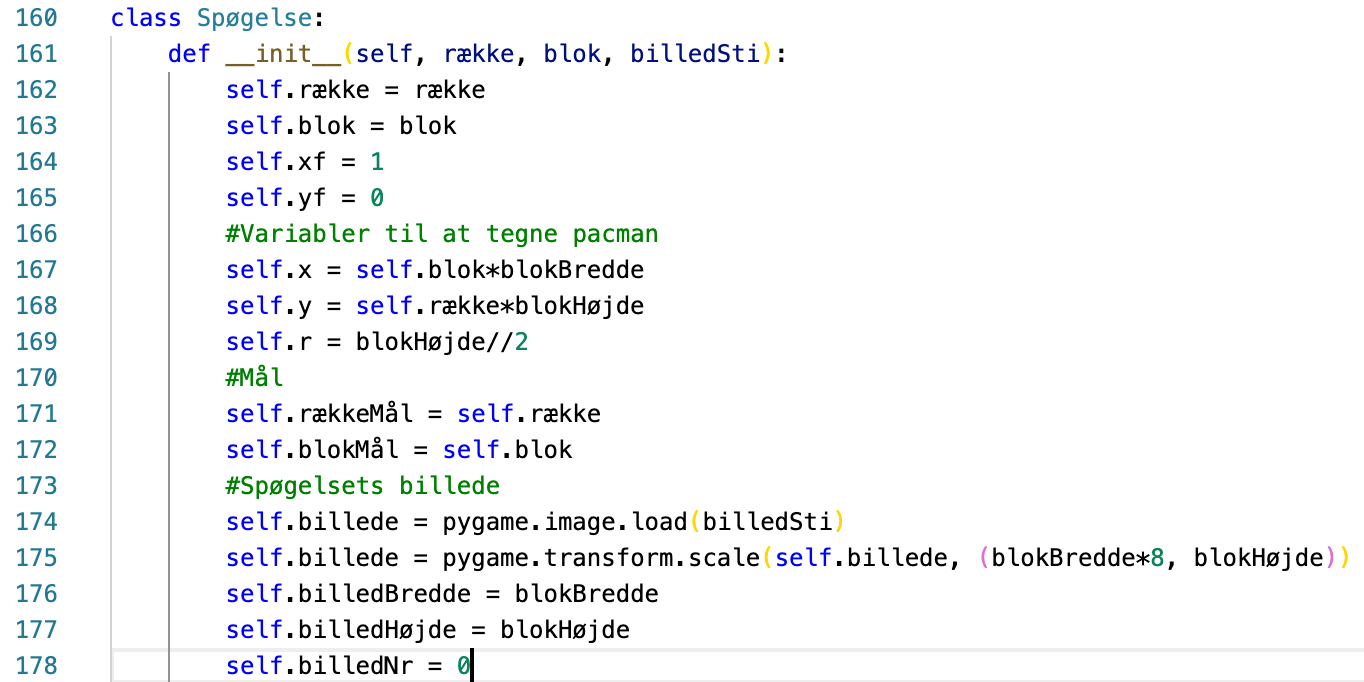
Tjek om det virker!

# Step 20: Spøgelserne skal også ligne spøgelser

Ja, de skal også have en sprite og den skal også bevæge sig. Lad os starte som før med at se på hvordan vi har spøgelsernes billeder. Herunder er Blinky spritesheet.



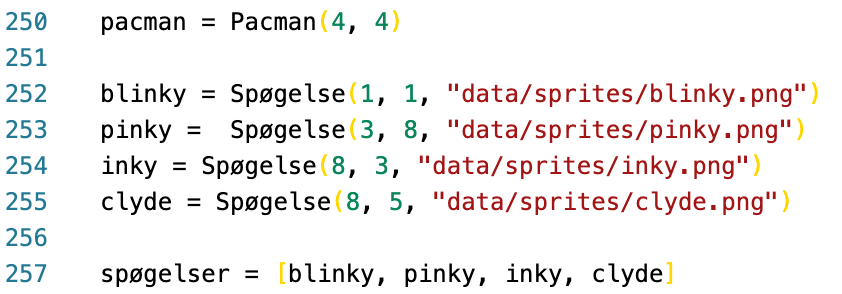
Vi ser her at spøgelsernes billede kun har en række, men med otte billeder svarende til to billeder pr. retning. Vi ser også at de to første billeder er spøgelsets der går op, det næste er spøgelset der går ned, så højre og så venstre. Ok lad os starte med bare at få vist et billede i stedet for spøgelset cirkel. For at vi kan det skal vi give hvert spøgelse deres eget billede (da de jo har forskellige farver), i stedet for at vi så har farve argumentet til vores Spøgelses \_\_init\_\_ funktion laver vi det nu om til billedets sti (altså hvor på computeren billedet er). Så starten af vores spøgelses klasse ender med at se sådan her ud:



Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseNu skal vi så bare lave vores tegn funktion om så den bruger billedet. Vi bruger samme funktion som i pacmans tegn funktion, så spøgelsets tegn funktion ender med at se således ud.

Nu mangler vi bare at give spøgelserne deres respektive billede. Så der hvor vi laver spøgelserne giver vi dem en sti så det ender med at se således ud:



Prøv at kør det nu ligner spøgelserne spøgelse?

# Step 20.1: Nu skal spøgelserne vende den rigtige vej

Ja, nu skal vi sørge for at spøgelserne også vender den rigtige vej. Som vi så på deres spritesheet kom de forskellige retninger lige efter hinanden og ikke i rækker. Vi skal derfor bare flytte billedNr hen på det rigtige billede. Så vi tjekker bare i tegn funktionen hvilken vej vores spøgelse vender og sætter billedNr.’et til den tilsvarende værdi. Så altså nul hvis det går ned, to hvis det går op, fire hvis det går til venstre og 6 hvis det går til højre. Så vores spøgelses tegn funktion ender med at se således ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseNu skulle spøgelserne gerne vende den rigtige vej.

# Step 20.2: Nu skal spøgelse så også lige animeres

Nu har vi billedet på spøgelserne og vi har sørget for at de vender den rigtige vej, nu mangler vi bare lige at få dem til at bevæge sig. Da vi kun har to billeder for hver retning kan vi gør lige som med pacman hvor vi har en variabel animer der skifter mellem 1 og 0 som vi så bare lægger til vores billedNr. Så vi starter med at definere animer variablen i spøgelsets \_\_init\_\_ funktion:

Et billede, der indeholder objekt

Automatisk genereret beskrivelse

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelseNu sørger vi bare for at animer variablen skifter mellem 1 og 0 i tegn og så lægger vi den til billedNr. Så spøgelsets tegn funktion ender med at se således ud:

Sådan nu skulle spøgelsernes billede også gerne ”bevæge” sig.

# Step 21: Hvad med det blå-spøgelse og øjnene?

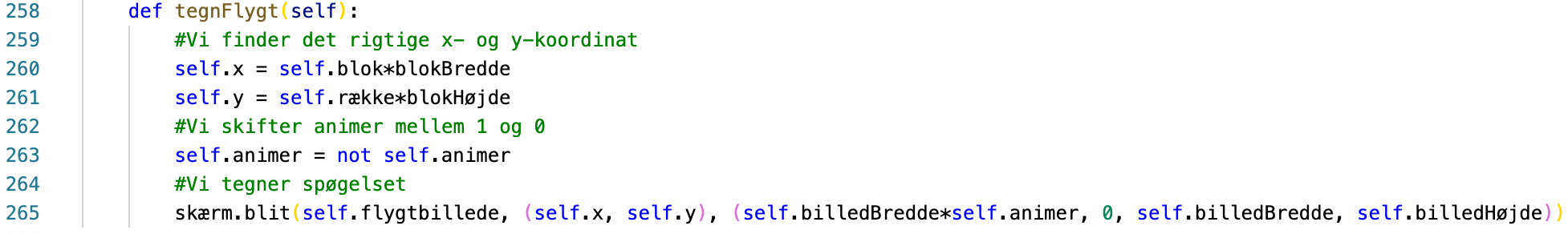
Nu hvor vi har fået tegnet spøgelserne når de bare ”tøffer” rundt, mangler vi stadig de to andre modes spøgelserne kan være i, de kan nemlig også flygte og være spist, hvor de henholdsvis skal være blå og bare øjne. Lad os starte med når spøgelserne flygter. Vi har allerede en tegnFlygt funktion i vores spøgelses klasse så lad os prøve bare at lave om på den. Hvis vi kigger i vores sprite mappe i data mappen er der en der spøgelseFlygt. Det er det billede vi gerne vil bruge, så lad os kigge på hvordan det spritesheet er struktureret.



Vi ser der er en række med fire kolonner. I denne omgang vil vi kun bruge de to første billeder. Da billedet deres bruges ved spøgelsernes flygt er det samme for dem alle sammen så kan vi bare skrive selve stien ind i \_\_init\_\_ funktionen ligesom med pacmans billede. Vi behøver heller ikke at tage højde hvilket billede vi er på da vi kun bruger de to første billeder kan vi bare bruge animer variablen fra steppet før. Vores spøgelses \_\_init\_\_ funktion ender med at se sådan her ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

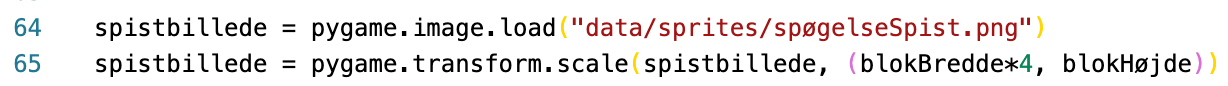
Nu skal vi så bare lave om på vores spøgelses tegnFlygt funktion så den bruger billedet, så den ender med at se således ud:

Så skulle spøgelserne gerne bruge flygt billede når de er flygt mode, prøv at se.

Hvis de bruger det rigtige billede i den mode så kan vi gå videre til at få øjnene med i spillet. Lad os starte med at se på hvordan det spritesheet ser ud.



Her ser vi at det består af en række med fire kolonner, hvor hver af disse er en forskellig retning. Ok, lad os så få dette billede ind i vores program. Da vi tegner de spiste spøgelser udenfor spøgelsesklassen i tegnSpisteSpøgelser funktionen skal vi sørger for at billedet er udenfor spøgelses klassen. Så oppe ved farverne kan vi nu indsætte følgende variabler:



Nu skal vi så bare bruge billedet i vores tegnSpisteSpøgelser funktion. Vi skal dog også lige sørger for at vi vælger det spøgelse der vender den rigtige vej. Vi starter derfor med at se på hvilken vej spøgelset bevæger sig og vælger billedet ud fra dette. Hvor hvis spøgelset bevæger sig opad bruger vi billede , hvis det bevæger sig ned billede 2, hvis det bevæger sig til venstre billede 3 og hvis det bevæger sig til højre billede 4. Så tegnSpisteSpøgelser ender med at se sådan her ud:

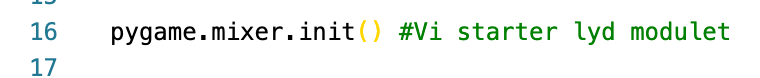
Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

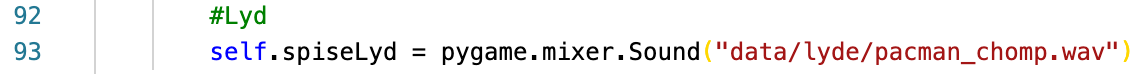
Nu skulle spøgelserne gerne have de rigtige øjne når de bliver spist.

# Step 22: LYD?

Nu mangler vi vist kun lidt noget lyd. Da jeg gerne vil have jer til at udfolde jeres kreativitet viser jeg kun hvordan man får spise lyden ind i spillet, de resterende lyde må i selv få ind. Så for at vi kan få lyde ind i pygame skal vi først have tændt for det der håndtere lydene, hvilket kaldes en mixer. Det gør vi ved at initiere det i starten, deroppe hvor vi laver vores skærm skriver vi så følgende:



Nu kan vi så begynde at afspille lyde. Da det i første omgang kun er pacmans spiselyd laver vi bare en lydvariabel i pacmans klasse som vi så afspiller hver gang han spiser en pille/ost. Så i pacmans \_\_init\_\_ funktion tilføjer vi følgende variabel. Denne funktion loader bare den lydfil ind der er på den givne sti.



Vi kan derefter afspille denne lyd nederst i pacmans flyt funktion, der hvor vi tjekker om han har spist en pille. For at undgå at den bliver afspillet super mange gange så spørger vi om den ikke bliver spillet i forvejen, hvis dette ikke er tilfældet så afspiller vi den. Så den nederste del af pacmans flyt funktion ender med at se således ud:

Et billede, der indeholder skærmbillede

Automatisk genereret beskrivelse

Så skulle den gerne afspille lyden.

# Step 23: Mangler der noget?

Nu har jeg ikke mere at byde på, men har du nogle idéer for så vil vi meget gerne hjælpe med dem. Det kunne f.eks. være at få flere lyde, et nyt spøgelse eller en ny bane (som er langt nemmere at gøre end i tror, jeg har nemlig snydt lidt.) Men overvej om der er noget andet du gerne vil have dit spil til at kunne, hvis du ikke kan komme på noget så spørg nogle af de andre og se om det ikke giver nogle gode idéer.