*Carl Christian David Dam*

*Sukkertoppen | 2.X*

*Flying Space Birds*

PROGRAMERINGSSYNOPSIS/JOURNAL



Indholdsfortegnelse

[Indledning 1](#_Toc133416449)

[Funktionsbeskrivelse 1](#_Toc133416450)

[Canvas-layout 1](#_Toc133416451)

[Styring 1](#_Toc133416452)

[Dokumentation af programmet 2](#_Toc133416453)

[State 2](#_Toc133416454)

[Baggrund 2](#_Toc133416455)

[Rør 3](#_Toc133416456)

[Test af program 3](#_Toc133416457)

[Bilag. 4](#_Toc133416458)

## Indledning

Jeg vil i mit eksamensprojekt udvikle et spil. Min umiddelbare tanke er noget i stil med *Space Invaders* og *Flappy Bird*, dog uden nogensinde at have spillet *Space Invaders*. Som udgangspunkt er det en raket (rektangel) som man styrer fra side til side. Det kan enten være med musen eller piletasterne. Udfordringen er så at der ”falder” grønne rør ned i hovedet på en, som man så skal undgå at bliver ramt af.

## Funktionsbeskrivelse

Canvas-layout:

Igennem undervisningsforløbet har jeg arbejder med P5’ standart (400,400) canvas, men har også eksperimenteret med andre størrelser og forhold. Senere har jeg også forsøgt mig med ”createCanvas(displayWidth, displayHeight);”

Til dette projekt valgte jeg tidligt i arbejdsprocessen at gå med (600,600) for at have lidt mere plads at arbejde med. Man ville godt kunne ændre den til noget større, men da mange variabler bygger på *width* ville man nok ende i nogle situationer hvor at man simpelthen ikke ville kunne nå igennem rørene, da ens horisontale hastighed er fast (x pixels per frame). Her ville man så skulle lave et forhold, så at det passer med bredden af canvas.

Styring:

Jeg havde forskellige muligheder for hvordan man skulle styre ”rumskibet”. Jeg kunne enten gøre det ud fra *mouseX* eller styre det med piletasterne, og valgte bare at bruge den sidste løsning. Man kunne også have valgt at man skulle bevæge sig som i *Flappy Bird,* men det ville ikke helt give mening efter som at alt er drejet 90 grader.

I forhold ti ”laseren” har jeg valgt at man både kan bruge pil op eller mellemrumstasten. Det følte jeg ville være ret naturligt, og så var der også noget at vælge imellem.

## Dokumentation af programmet

State:

Mit menu-system er sat op, sådan at hver ”menu” (det inkluderer også selve spillet) er et *State.* Hver *State* har så en værdi. State er altså bare en variable float.

|  |  |
| --- | --- |
| State-tabel | Værdi: |
| Hovedmenu | 0 |
| Selve spillet | 1 |
| Vindermenu | 2 |
| How to play | 3 |

Her ville jeg så fx bare kunne lave en ny *State*:

State = 4;

Det er så forskellige events det skifter mellem *States*. I menuerne er det hvis man trykker på en knap, og i spillet er det når man er færdig.

Function draw() {

If (state ==0) {

MainMenu()

}

If (state ==1) {

Game()

}

}

If (button == pressed) {

State = 1

}

//Eller:

If (game == done) {

State = 0

}

Baggrund:

Min baggrund består af 200 cirkler, der skal forestille at være stjerner. For at lave dem har jeg to funktioner. Den første generateStars, tager et parameter ind, som er hvor mange stjerner der skal være.

generateStars kører så en for-lykke igennem, og genererer stjerner med et x- og y-koordinat, og en radius.

Den anden funktion updateBG, bliver kaldt hver frame, og står for at stjernerne ikke er statiske.

Først laver den faktisk en baggrund, med P5’s egen funktion. Her har jeg valgt en farve som ikke er helt sort, så man ikke får trætte øjne af at kikke på det i længere tid.

Herefter er det så at jeg faktisk tegner alle cirklerne med en næsten-hvid farve, samme grund som før.

Bagefter går jeg så igen alle cirklerne igennem igen, for at ændre på deres hastighed, det vil sige at jeg lægger noget til deres y-værdi. For at finde ud af hvad hastigheden skal være, bruger jeg deres radius i en map-funktion. Det vil sige at jo mindre en stjerne er, jo lavere vil dens hastighed også være. Og da starMaxSpeed er mindre end rørenes hastighed, vil man også få en effekt af at de faktisk er i baggrunden.

Det sidste der sker i updateBG er at den (igen) går alle stjernerne igennem, men nu for at tjekke om de er på skærmen, og hvis de ikke er det, bliver de rykket op lidt over ”canvas”, så det ikke ser underligt ud.

### Rør

Måden jeg laver rørene på, er i virkeligheden ved at, i stedet for at lave to rør ved siden af hinanden, så ”spiser” jeg en del af røret, så der kommer et hul. På den måde kan jeg variere positionen og bredden af hullet, så at det er nyt hver gang der kommer et nyt rør. Det giver mig også nogle variabler, som jeg så bruger til at udregne kollision mellem rørene og spilleren.

## Test af program

Jeg har løbende kørt koden, mens jeg har skrevet den. Det har gjort at jeg med det samme viste hvad jeg har ændret, og hvad der derved kan fejle i programmet.

Der er nogle buggs i programmet, som jeg godt er klar over, fx burde der være en variabel, for hvornår et man gik i gang med spillet, så altså når man klikker på PLAY, i stedet for at den bare tæller millis.

## Bilag.

function updateBG() {

background(22, 20, 22);

fill(230, 230, 230);

noStroke;

for (let i = 0; i < starpossition.length; i++) {

circle(starpossition[i][1], starpossition[i][2], starpossition[i][0]);

}

for (let i = 0; i < starpossition.length; i++) {

starpossition[i][2] =

starpossition[i][2] +

map(starpossition[i][0], 0, starMaxR, 0, starMaxSpeed);

}

for (let j = 0; j < starpossition.length; j++) {

if (starpossition[j][2] > height + 10) {

starpossition[j][2] = -40;

}

}

}