SYNOPSIS

Eksamensprojekt i programmering

Matias Holst & Thomas Askov Søndergaard
Matiasholst787@gmail.com & thoaskson@gmail.com

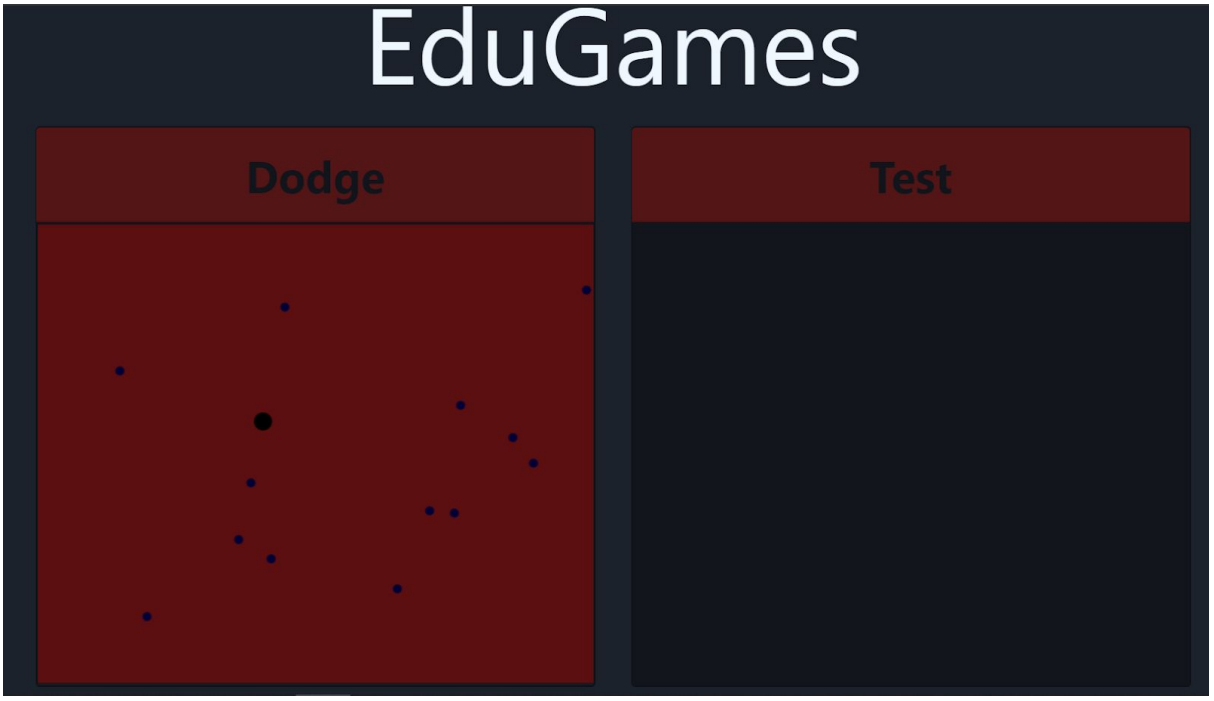
Resume

Denne opgave er skrevet i forbindelse med afslutningen af faget programmering B i 2.G. Det er blevet lavet som værende eksamensprojekt. Programmet er lavet i p5.js, som er et visuelt kodningsværktøj, og HTML, hvis formål er at få tekst, billeder og programmer til at fungere ordentligt på hjemmesider.

Programmet er en hjemmeside, som kan føre dig ind på spil, hvis mål er at uddanne brugeren på en sjov måde. Dette gøres gennem brug af et spil, hvor spilleren skal svare på et matematikspørgsmål, når han/hun dør,

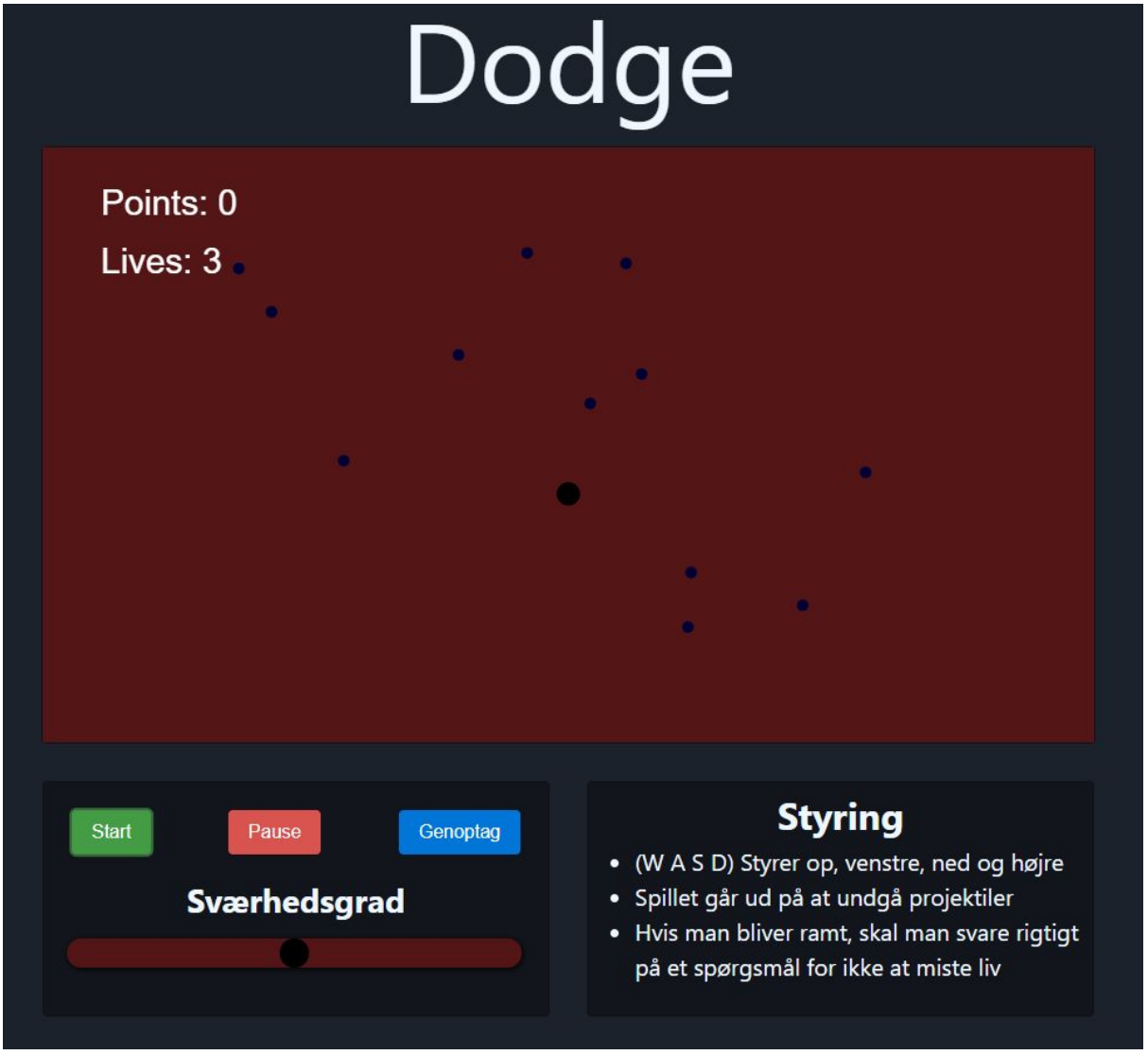
Programmet er målrettet efter brug i undervisningen af de yngre klassetrin, da det kan gøre det sjovere at træne additions-, subtraktions- og multiplikationsstykker.

Nedenunder ses et skærmbillede fra 'hjemmeside'-delen af programmet:



Trykkes der på billedet tilhørende 'Dodge', kommer man ind på spillet 'Dodge'

Nedenunder ses et skærmbillede fra 'Dodge'-delen af programmet:



Problemformulering

Når unge elever skal lære ting såsom simpel hovedregning, kan det være meget svært at motivere dem. Det gør, at de senere i livet kan opleve problemer i forbindelse med sværere matematik, da de mangler et fundament at bygge videre på. Derfor er det en rigtig god idé at gøre noget for at læringen er sjov og uden den store anstrengelse. Dette kan f.eks. gøres gennem en hjemmeside med lærerige computerspil. Der opstår et par problemer og overvejelser under udviklingen af sådan et spil, såsom:

- Hvordan kan man motivere børn til at lære vha. programmering?
- Hvordan kan man kommunikere programmets styremetoder til brugeren på en intuitiv måde?
- Hvordan kan man få et spil til at fungere som forlængelse af en hjemmeside?

- Hvordan kan man håndtere sådan et program på en objektorienteret måde?

Metoder

- **Agile vs. Vandfald**

  Når et program skal udvikles, er det vigtigt, at man allerede har en idé om hvordan man skal gå i gang med arbejdet for at opnå en så problemfri arbejdsproces som muligt. Til dette er der hovedsageligt tale om to forskellige arbejdsmetoder: "Agile" og "Vandfald".

  Når der gøres brug af "Agile"-metoden, vil det sige, at man gennem arbejdsforløbet løbende finder ud af, hvad der skal tilføjes til programmet for at skabe et godt endeligt produkt. Det vil sige, at man går ind i arbejdsforløbet med en generel idé om, hvordan det endelige program skal se ud, og hvad det skal kunne men uden en uforanderlig plan om hvordan det endelige resultat skal opnås. Fordelen ved denne måde at programmere på, er at arbejdsforløbet bliver mere fleksibelt, og at der, hvis man gennem forløbet kommer på gode idéer til tilføjelser til programmet, er mulighed for at indføre disse. Ulempen er dog, at man, modsat med "Vandfald"-metoden ikke i så høj grad har en fast plan, hvilket kan føre til ustruktureret arbejde og forvirring over, hvad der skal gøres som det næste.

  Når der gøres brug af "Vandfald"-metoden, vil det sige, at man, allerede før man begynder at skrive selve koden, både har en helt klar vision om, hvordan programmet skal ende med at være, og at man har en konkret plan om, hvad der skal gøres fra start til slut i arbejdsforløbet. Fordelen ved dette er, at man under arbejdsforløbet aldrig behøver at stille spørgsmål til, hvad det næste, der skal gøres, er. Det vil sige, at alle parter involveret i programmets skabelse vil kunne arbejde målrettet indtil produktet er fuldendt, uden større diskussion om hvad programmet skal kunne, efter man for første gang er blevet enige om dette.

  Vi har under udviklingen af vores program gjort brug af "Agile"-metoden. Dette har vi gjort, fordi vi på forhånd vidste, at programmet ville blive et relativt stort program, hvori der skulle anvendes flere forskellige filer, der alle skulle kommunikere med hinanden. Af denne grund var det mere praktisk for os, at vi, gennem arbejdsforløbet, kunne gå tilbage og ændre ting i programmet, som vi mente ville fungere bedre på en anden måde. Desuden havde vi før skabelsen af programmet ikke et helt konkret billede i hovedet af, hvordan det endelige program ville komme til at se ud. Vi havde en generel idé om dette, men de finere detaljer faldt det os mere naturligt at snakke om finde ud af hen ad vejen.

- **Objekt orienteret analyse (OOA) og objekt orienteret design (OOD)**

  Når man laver et stort program, kan det være en rigtig god idé at anvende objektorienteret programmering, da der er mange forskellige ting i programmet, der minder meget om hinanden. Af den grund skaber man såkaldte 'klasser', altså generelle idé om, hvad den type objekter, der hører til klassen er og kan. Ud fra disse klasser skaber man 'instanser', altså at man 'vækker klasserne til live'. Disse instanser giver man nogle bestemte værdier, så man ender med forskellige instanser fra samme klasse, der alle har de samme egenskaber men med forskellige attributter.

  I vores program har vi gjort brug af objektorienteret programmering, når vi skaber vores spiller og de projektiler, som spilleren skal undvige. Disse klasser har vi kaldt henholdsvis "Player" og "Enemy".

  Selvom der kun kreeres én instans af klassen "Player", giver det god mening at give den en klasse for sig selv, da spilleren på den måde kan refereres til og behandles som et objekt gennem resten af programmet. Desuden kan vi på denne måde også give spilleren, "Player", attributter såsom spillerens liv, som hører særligt til denne klasse og kun denne klasse.

  Det er mere åbenlyst, hvorfor det er smart at lave en klasse for fjenderne, "Enemy", da der skal laves mange forskellige fjender. Alle fjenderne har samme egenskaber, men de starter forskellige steder på skærmen og bevæger sig forskelligt med forskellige hastigheder. Af den grund skaber vi forskellige instanser, der hver har forskellige x- og y-værdier, hastigheder og bevægelseskurver.

- **Matematisk Modellering**

  I vores program har vi til bevægelsen af de fjendtlige projektiler gjort brug af matematisk modellering i form af tre forskellige funktionstyper: Lineære funktioner, andengradsfunktioner og trigonometriske funktioner

  De lineære funktioner er skabt ud fra udtrykket $y = ax + b$

  Andengradsfunktionerne er skabt ud fra udtrykket $y = ax^2 + bx + c$

  De trigonometriske funktioner er skabt ud fra udtrykket $y = a \cdot sin(bx \cdot c) + d$

  Alle variablerne til funktionerne bliver beregnet og indsat i funktionerne af programmet.

Programbeskrivelse, Funktionalitet, Brugergrænseflader, Udvalgt kode

```
move() {
  if (keyIsDown(68)) {
    this.xPos += this.xSpeed;
  }
  if (keyIsDown(65)) {
    this.xPos -= this.xSpeed;
  }
  if (keyIsDown(87)) {
    this.yPos -= this.ySpeed;
  }
  if (keyIsDown(83)) {
    this.yPos += this.ySpeed;
  }
}
```

Programmets mest centrale del er selve spilleren, da det er denne som brugeren skal styre og prøve at undvige projektiler med. Spilleren kan bevæges på en let og intuitiv måde, og denne bevægelse sættes derfor på knapper w, a, s og d for henholdsvis, opad-, venstre-, nedad- og højrerettet bevægelse, da dette er en meget typisk kombination for disse. Disse knapper er alle skrevet ind med deres tilsvarende "keycodes". (Se billedet hvor funktionen "move" kan ses) Disse styrefunktioner kan ses i små kasser under spillet ved siden af en kasse, hvor der kan vælges sværhedsgrad, og hvor der er knapper til at starte, pause og genoptage spillet. Alt dette kommunikeres klart til brugeren gennem en GUI, der tydeligt beskriver sine funktioner, både gennem "color coding" og reglerne om adskilthed og indelukkethed.

Programmet er et spil med målet at få så mange point som muligt. Spillet fungerer ved at projektiler bliver skudt mod spilleren fra både højre og venstre side af skærmen, hvorefter spilleren skal gøre sit bedste for ikke at blive ramt. Dette kan være besværligt da projektilerne bevæger sig med forskellige hastigheder og bevægelser i form af lineære-, andengrads- og trigonometriske funktioner, som sammen gør det svært at undvige. Pointtallet stiger når projektilerne forlader skærmen, men det samme gør antallet af projektiler. Jo mere man spiller jo sværere bliver spillet altså, men på grund af det større antal projektiler som forlader skærmen stiger pointtallet hurtigere. Et eksempel på en sådan beregning kan ses til højre (resten kan findes i bilag):

```
  if (projectileType == 3) {
    this.a = random(80, height / 2);
    this.b = random(0.001, 0.005);
    this.c = random(0, TWO_PI);
    this.d = height / 2 + random(-40, 40);

    return [a, b, c, d];
  }
}
```

Projektilernes bane skabes derefter ud fra forskellige matematiske udtryk, alt efter hvilket type projektil der er tale om. (Se afsnittet: "Matematisk Modellering")

```
function questionGen() {
    questionType = floor(random(1, 4));
    number1 = floor(random(1, 51));
    number2 = floor(random(1, 51));

    if (questionType == 1){
        result = number1 + number2;
    }

    if (questionType == 2){
        result = number1 - number2;
    }

    if (questionType == 3){
        number1 = floor(random(1, 26));
        number2 = floor(random(1, 26));
        result = number1 * number2;
    }
}
```

I tilfælde af, at man bliver ramt af et projektil, vil spillet blive sat på pause, og et spørgsmål vil stå på skærmen. Spørgsmålene er hovedpointen med spillet, da det er designet med undervisning som værende det primære formål. Disse spørgsmål varierer mellem additions-, subtraktions- og multiplikationsstykker og indeholder tal indenfor et bestemt interval, valgt af os. Alt dette ligger under funktionen "questionGen", altså funktionen der genererer spørgsmål.

Hvis man svarer rigtigt på et spørgsmål, vil spillet køre videre som normalt, hvorimod hvis man svarer forkert mister man et liv, hvorefter spillet starter igen. Der bliver hele tiden tjekket, hvor mange liv man har gennem funktionen "checkDie", og når man rammer 0 liv, vil der komme en GUI op på skærmen, der spørger om man vil prøve igen eller ej. Dette gøres på en måde, der effektivt kommunikerer til brugeren, at de døde, og hvad de nu skal gøre.

```
function checkDie (player){
    if (player.playerLife == 0){
        noLoop();
        clear();
        background(83, 21, 22);
        fill((255, 255, 255));
        textSize(30);
        textAlign(CENTER, CENTER);
        text("Du tabte. Vil du prøve igen?", width/2.05, height/2.5);

        tryAgain = createButton('Ja');
        tryAgain.position(windowWidth * 0.75, height / 1.5);
        tryAgain.mousePressed(playAgain);

        goBack = createButton('Nej');
        goBack.position(windowWidth * 0.25, height / 1.5);
        goBack.mousePressed(goToFrontpage);
    }
}
```

Hele programmet kommunikerer med og kører over en hjemmeside, nemlig hjemmesiden "EduGames", som vi selv har lavet. På denne hjemmeside kan man vælge at spille spillet 'Dodge', altså det førnævnte spil, og det kan også ses, at der er plads til og mulighed for indførsel af flere spil, skulle dette ønskes. Trykkes der på billedet af "Dodge", sendes man videre til spillet og kan frit spille, til man ikke ønsker det mere. Derudover er der inde på spillet en "home"-knap, der, når der trykkes på denne, sender brugeren tilbage til forsiden, hvor der kan vælges et andet spil, hvis disse var indsat.

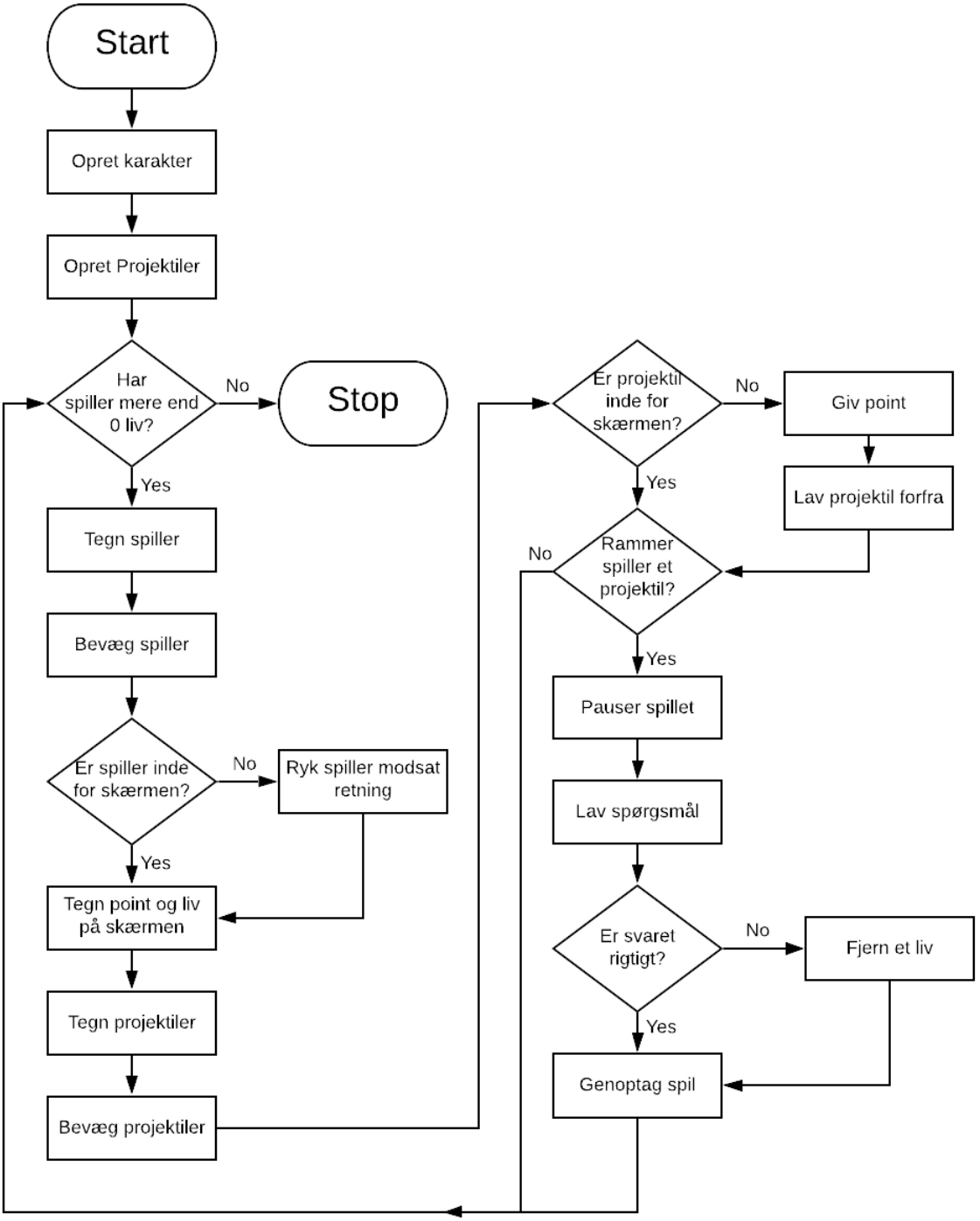
Pseudokode

Programmet, Dodge, virker på følgende måde:

- Opret spiller og liste af projektiler
- Giv projektilerne tilfældige egenskaber fx. startposition, hastighed, bevægelsestype.
- Knapperne WASD styrer spilleren op, venstre, ned, højre.
- Bevæg projektilerne hen over skærmen mod den modsatte side vha. bevægelsestypen fx lineær, andengrads, trigonometriske
- Når projektilerne rammer siderne af skærmen laves de forfra med nye variabler og spilleren får tilføjet point.
- Hvis spilleren bliver ramt af et projektil skal spillet pauses, hvorefter et spørgsmål gives
- Svares rigtigt kører spillet videre. Svares forkert mister spilleren et liv
- Når spillerens liv når 0 stopper spillet. Programmet tjekker om man trykker på "Ja" eller "Nej" til spørgsmålet "Prøv igen?". Hvis "Ja", genindlæs hjemmeside, hvis "Nej", gå til forside
- Spillet kan pauses og genstartes på ethvert tidspunkt vha. knapper

Flowchart

Dodge fungerer som illustreret på følgende flowchart:



Class diagram

Class: Player	**Class: Enemy**
Attributter: - this.xSpeed - this.ySpeed - this.xPos - this.yPos - this.diameter - this.radius - this.color - this.playerLife	Attributter: - this.speed - this.diameter - this.radius - this.projectileType - this.function - this.x - this.y - this.hasCollided - this.startPosition
Metoder: - move() - display() - checkXY() - isColliding()	Metoder: - display() - move() - startFromOtherSide() - wallCollision()

Class beskrivelse

Class: Player
Klassen “Player” opretter hovedspilleren som kan bevæges med WASD knapperne. Der bliver sikret at spilleren ikke kan flyttes ud af skærmen og samtidig tjekket om projektiler rammes. Karakteren får nogle variabler som bestemmer hastighed, startposition, størrelse og farve. For at gøre det, som klassen skal kunne, har den desuden nogle tilhørende metoder, der alle gør forskellige ting, som giver klassens dens funktionaliteter.

Class: Enemy
Klassen “Enemy” opretter et cirkulært objekt som ved hjælp af metoden move() bevæger sig fra enten venstre mod højre eller højre med venstre, så den ligner et projektil. Klassen bruges til at oprette flere af disse projektiler for at øge spillets sværhedsgrad. Objektet får ligesom “Player” nogle variabler som bestemmer dens start position, hastighed og type “projectileType”. Klassen giver projektilerne en type som bestemmer deres retning og bevægelses måde. Når objektet remmer enten højre eller venstre side af skærmen bliver den lavet helt forfra med nye tilfældige variabler.

Funktionsbeskrivelse

setup(): kører som det første i programmet og laver nogle variablers startværdier og objekter. setup() kører kun én gang
draw(): kører 60 gange i sekundet. Her opdateres spiller, projektil position og generelt hele programmet. Draw er på en måde den vigtigste funktion da det visuelle er meget vigtigt for spillet.
move(): bruges til at ændre x og p position for spilleren og projektilerne
display(): tegner spilleren og projektilerne
checkXY(): tjekker om spilleren bevæger sig ud af skærmen og stopper dem hvis de rammer kanten
isColliding(): sammenligner distancen mellem projektil og spiller og tjekker om de rammer
getFunction(): udregner koefficienterne, der skal bruges til at udregne y alt efter projektiltypen. Altså a og b til linær, a, b og c til andengrads og a, b, c og d til trigonometrisk.
getY(): udregner y, altså funktionen som projektilerne bevæger sig efter, ud fra projektiltypen.
startFromOtherSide(): laver halvdelen af projektilerne på den anden side af skærmen for at øge sværhedsgraden.
wallCollision(): tjekker om projektilerne remmer siderne på skærmen og skaber dem på ny med nye variabler.
run(): kører spillet

questionGen(): generer et tilfældigt additions, subtraktions eller multiplikationsstykke
questionScreen(): stopper spillet og får et spørgsmål frem på skærmen, når man bliver ramt af et projektil
answer(): tjekker om svaret er det samme som resultatet af spørgsmålet. Hvis svaret er forkert mister spilleren et liv. answer() starter også spillet igen efter man har svaret
GUI(): viser point og liv i øverst venstre side af skærmen
playAgain(): genindlæser siden, så spillet kan startes forfra
goToFrontpage(): er en knap som sender brugeren til hjemmesidens forside
checkDie(): tjekker om spilleren har flere liv tilbage. Hvis spilleren rammer 0 liv stoppes spillet, og man bliver spurgt om man vil prøve igen.

Test

Spillet er kun blevet testet i 'Google Chrome'-browseren, og det kan dermed som sådan ikke sige om det fungerer optimalt på andre browserer. Derudover er det blevet opdaget, at der er problemer med positioner af tekstbokse på forskellige skærmstørrelser. Problemet med forskellige skærmstørrelser skulle kunne løses ved at gøre tekstboksenes positioner afhængige af skærmstørrelsen, som programmet bliver kørt på.

Udover disse problemer virker programmet til at fungere, som det skal.

Konklusion

Det kan konkluderes, at det gennem en hjemmeside og et spil, såsom 'Dodge', er muligt at inkorporere sjov og spil som en del af læring af f.eks. matematik.

Da det er relativt små børn, spillet er lavet til, er det ekstra vigtigt, at det er nemt og intuitivt at spille spillet og finde rundt i det. Dette gøres vha. en klar GUI, der viser, hvad slideren 'sværhedsgrad' gør. hvad de tre farvede knapper gør og en boks, hvor der står, hvordan man styrer spillet.

Ved at gøre spillet til en del af en hjemmeside åbner det muligheden for at på længere sigt kunne tilføje flere spil, da der allerede er en central for spillene. Derudover gør det det meget intuitivt for brugeren at kunne finde ind på spillene, da langt de fleste i vores målgruppe, altså børn, allerede kender hjemmesider som vores, hvilket gør, at de fra de første gang ser hjemmesiden kan navigere den.

Det giver rigtig god mening at håndtere programmet på en objektorienteret måde ved at gøre spilleren og fjenderne til forskellige klasser. Dette giver rig mulighed for at håndtere dem separat men alligevel få dem til at interagere med hinanden i programmet, samtidig med at det giver mulighed for at skabe så mange fjender som man vil uden større problemer, så snart klassen er blevet oprettet.

Alt i alt er det lykkedes at lave et program, der opfylder de tidligere opstillede krav til løsningen af vores problemformulering.

Bilag

Kode med ”style” og kommentar

Dodge.js

```
/*jshint esversion: 7 */

var player;

//  Skaber en klasse for spilleren
class Player {
  constructor(xPos, yPos, diameter) {
    this.xSpeed = 3;
    this.ySpeed = 3;
    this.xPos = xPos;
    this.yPos = yPos;
    this.diameter = diameter;
    this.radius = this.diameter * 0.5;
    this.color = (0);

    // Giver spilleren et antal liv
    this.playerLife = 3;
  }

  //  Bevæger spilleren når forskellige knapper trykkes ned
  move() {
    if (keyIsDown(68)) {
      this.xPos += this.xSpeed;
    }
    if (keyIsDown(65)) {
      this.xPos -= this.xSpeed;
    }
    if (keyIsDown(87)) {
      this.yPos -= this.ySpeed;
    }
    if (keyIsDown(83)) {
```

```
      this.yPos += this.ySpeed;
    }
  }

  // Tegner spilleren
  display() {
    let c = color(0, 0, 0);
    fill(c);
    circle(this.xPos, this.yPos, this.diameter);
  }

  // Checker om spilleren bevæger sig uden for skærmen
  checkXY() {
    if (this.xPos - this.radius <= 0) {
      this.xPos += this.xSpeed;
    }
    if (this.xPos + this.radius >= width) {
      this.xPos -= this.xSpeed;
    }
    if (this.yPos - this.radius <= 0) {
      this.yPos += this.ySpeed;
    }
    if (this.yPos + this.radius >= height) {
      this.yPos -= this.ySpeed;
    }
  }

  // Tjekker om spilleren rammer en fjende
  isColliding(enemy) {
    let distancePlayerEnemy = dist(this.xPos, this.yPos, enemy.x,
enemy.y);

    if (distancePlayerEnemy < this.radius + enemy.radius &&
enemy.hasCollided == false) {
      enemy.hasCollided = true;
      return true;
    }

    return false;
  }
}

// Opretter funktion for  beregning af forskellige typer projektilers
koefficienter
function getFunction(projectileType) {

  // Beregning af koefficienter til linær funktion
  if (projectileType == 1) {
```

```
    x1 = 0;
    y1 = random(0, height);
    x2 = width;
    y2 = random(0, height);
    slope = (y2 - y1) / (x2 - x1);
    intersection = y1;

    return [slope, intersection];
  }

  // Beregning af koefficienter til andendgradsfunktion
  if (projectileType == 2) {
    x1 = 0;
    y1 = random(0, height);
    x2 = width / 2;
    y2 = random(0, height);
    x3 = width;
    y3 = random(0, height);

    quadraticA = (((x2 - x1) * (x3 ** 2) + (-(x2 ** 2) + (x1 ** 2)) *
x3 + x1 * (x2 ** 2) - (x1 ** 2) * x2) ** -1) * ((x2 - x1) * y3 + (-x3 +
x1) * y2 + (x3 - x2) * y1);
    quadraticB = -(((x2 - x1) * (x3 ** 2) + (-(x2 ** 2) + (x1 ** 2)) *
x3 + x1 * (x2 ** 2) - (x1 ** 2) * x2) ** -1) * (((x2 ** 2) - (x1 ** 2))
* y3 + (-(x3 ** 2) + (x1 ** 2)) * y2 + ((x3 ** 2) - (x2 ** 2)) * y1);
    quadraticC = (((x2 - x1) * (x3 ** 2) + (-(x2 ** 2) + (x1 ** 2)) *
x3 + x1 * (x2 ** 2) - (x1 ** 2) * x2) ** -1) * ((x1 * (x2 ** 2) - (x1
** 2) * x2) * y3 + (-x1 * (x3 ** 2) + (x1 ** 2) * x3) * y2 + (x2 * (x3
** 2) - (x2 ** 2) * x3) * y1);

    return [quadraticA, quadraticB, quadraticC];
  }

  // Beregning af koefficienter til trigonometrisk funktion
  if (projectileType == 3) {
    this.a = random(80, height / 2);
    this.b = random(0.001, 0.005);
    this.c = random(0, TWO_PI);
    this.d = height / 2 + random(-40, 40);

    return [a, b, c, d];
  }
}

// Beregner forskellige projektilers baner
function getY(x, projectileType, mathFunction) {
  // Beregner banen for projektiler, der bevæger sig som en linær
funktion
```

```
  if (projectileType == 1) {
    return (mathFunction[0] * x + mathFunction[1]);
  }
  // Beregner banen for projektiler, der bevæger sig som en
andengradsfunktion
  if (projectileType == 2) {
    return (mathFunction[0] * (x ** 2) + mathFunction[1] * x +
mathFunction[2]);
  }
  // Beregner banen for projektiler, der bevæger sig som en
trigonometrisk funktion
  if (projectileType == 3) {
    return (mathFunction[0] * sin(mathFunction[1] * x *
mathFunction[2]) + mathFunction[3]);
  }
}

// Skaber en klasse for fjenderne
class Enemy {
  constructor(diameter, speed, projectileType) {
    this.speed = speed;
    this.diameter = diameter;
    this.radius = diameter * 0.5;
    this.projectileType = projectileType;
    this.function = getFunction(this.projectileType);
    this.x = 0;
    this.y = 0;
    this.hasCollided = false;
    this.startPosition = Math.round(Math.random());

    // Gør at fjenderne kommer fra højre hvis startPosition == 1
    if (this.startPosition == 1) {
      this.startFromOtherSide();
    }
  }

  // Tegner fjender
  display() {
    let c = color(0, 0, 55);
    fill(c);
    circle(this.x, this.y, this.diameter);
  }

  // Bevæger fjender
  move() {
    this.x += this.speed;
    this.y = getY(this.x, this.projectileType, this.function);
  }
```

```
  // Laver en funktion, der kan bruges til at få fjender til at komme
fra højre
  startFromOtherSide() {
    this.speed = -this.speed;
    this.x = width;
  }

  // Tjekker om fjender rammer kanten af skærmen
  wallCollision() {
    if (this.x < 0 || this.x > width) {
      return true;
    }

    return false;
  }
}

start = false;
points = 0;
counter = 0;

function setup() {
  // Skaber kanvas og får det ind på hjemmesiden
  canvas = createCanvas(900, 510);
  canvas.parent("canvas-holder");
  noStroke();

// Skaber spilleren
player = new Player(width / 2, height / 2, 20);

  //Starter spillet når der trykket på startGame
  document.getElementById('startGame').onclick = () => {
    // Bruger difficulty til at beregne antal af fjender
    difficulty = document.getElementById('difficultySlider').value;
    enemies = [];
    amountOfEnemies = difficulty * 2 + 2;
    enemySize = 10;

    //Skaber et antal fjender svarende til amountOfEnemies
    for (let i = 0; i < amountOfEnemies; i++) {
      append(enemies, new Enemy(enemySize, random(1, 6),
floor(random(1, 4))));
    }
    run();
  };

  // Pauser spillet når der trykkes på resumeGame
```

```
  document.getElementById('pauseGame').onclick = () => noLoop();
  // Genoptager spillet når der trykkes på resumeGame
  document.getElementById('resumeGame').onclick = () => loop();
}

// Opretter en funktion der kører spillet
function draw() {
  checkDie(player);

  // Tegner baggrund
  background(83, 21, 22);

  // Tegner og bevæger spilleren og sørger for at spilleren ikke ryger
udenfor skærmen
  if (start) {
    player.display();
    player.move();
    player.checkXY();

    GUI(points, player.playerLife);


    for (let i = 0; i < enemies.length; i++) {
      enemies[i].move();
      enemies[i].display();

      if (enemies[i].wallCollision()) {
        points += 50;
        counter += 1;
        if (counter == 5) {
          enemies[enemies.length] = new Enemy(enemySize, random(1, 5),
floor(random(1, 4)));
          counter = 0;
          console.log(enemies.length);
        }
        enemies[i] = new Enemy(enemySize, random(1, 5), floor(random(1,
4)));
      }

      if (player.isColliding(enemies[i])) {
        document.getElementById('resumeGame').disabled = true;
        document.getElementById('startGame').disabled = true;
        questionGen(floor(random(1, 4)));
        questionScreen();
      }
    }
  }
  checkDie(player);
```

```
}
```

Functions.js

```
/*jshint esversion: 7 */

// Funktion der kører spillet når der trykkes på start
function run() {
    start = true;
}

// Funktion der tjekker om svaret er rigtigt og trækker et liv fra
spilleren hvis det ikke er
function answer() {
    ans = input.value();

    console.log(str(ans), result);

    if (float(ans) == float(result)) {
        loop();
    }
    else {
        player.playerLife -= 1;
        loop();
        console.log(player.playerLife);
    }

    input.hide();
    button.hide();
    document.getElementById('resumeGame').disabled = false;
    document.getElementById('startGame').disabled = false;
}

// Funktion der får et spørgsmål frem på skærmen når man bliver ramt
function questionScreen (){
    questionGen();
    noLoop();

    if (questionType == 1) {
        type = " + ";
    } else if (questionType == 2) {
        type = " - ";
    } else if (questionType == 3) {
        type = " · ";
    }
    fill((255, 255, 255));
    textSize(30);
    textAlign(CENTER, CENTER);
```

```
    text("Hvad er " + str(number1) + str(type) + str(number2) + "?",
width/2.05, height/2.5);

    input = createInput();
    input.id = "guess";
    input.position(windowWidth/2.45, windowHeight/3.5);

    button = createButton('Svar');
    button.position(input.x + input.width, input.y);
    button.mousePressed(answer);
}
// Funktion der genererer tilfældige matematik spørgsmål
function questionGen() {
    questionType = floor(random(1, 4));
    number1 = floor(random(1, 51));
    number2 = floor(random(1, 51));

    if (questionType == 1){
        result = number1 + number2;
    }

    if (questionType == 2){
        result = number1 - number2;
    }

    if (questionType == 3){
        number1 = floor(random(1, 26));
        number2 = floor(random(1, 26));
        result = number1 * number2;
    }
}
// Funktion som viser liv og point
function GUI(points, lives) {
    fill((255, 255, 255));
    textSize(30);
    textAlign(LEFT, CENTER);
    text("points: " + str(points), 50, 50);
    text("Lives: " + str(lives), 50, 100);
}


function playAgain(){
    location.reload();
}

function goToFrontpage(){
```

```
window.location.replace("https://bigchungus420.github.io/EduGames/src/index.html");
}

// Funktion der tjekker om man er død
function checkDie (player){
    if (player.playerLife == 0){
        noLoop();
        clear();
        background(83, 21, 22);
        fill((255, 255, 255));
        textSize(30);
        textAlign(CENTER, CENTER);
        text("Du tabte. Vil du prøve igen?", width/2.05, height/2.5);

        tryAgain = createButton('Ja');
        tryAgain.position(windowWidth * 0.75, height / 1.5);
        tryAgain.mousePressed(playAgain);

        goBack = createButton('Nej');
        goBack.position(windowWidth * 0.25, height / 1.5);
        goBack.mousePressed(goToFrontpage);
    }
}
```

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" type="text/css" href="style.css">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css"

integrity="sha384-rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ" crossorigin="anonymous">

    <title>EduGames</title>
```

```
</head>

<body class="body">
    <!-- Overskrift -->
    <div style="text-align: center;">
        <h1 class="display-1" style="color:
aliceblue;"><b>EduGames</b></h1>
    </div>

    <!-- Her laves kasser til forskellige spil -->
    <div class="container-fluid">
        <div class="row justify-content-center" id="space-between">

            <!-- Kasse til "Dodge" spillet  -->
            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Dodge</b>
                    </div>
                    <div class="card-body" id="card-body">
                        <!-- Link til "Dodge" hjemmeside -->
                        <a
href="https://bigchungus420.github.io/EduGames/src/Dodge.html"
class="image" title="Dodge Game"
                        style="position: relative; left:1px; top:1px;">
                            <img src="images\Dodge_498x413.png">
                        </a>
                    </div>
                </div>
            </div>

            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">

                    </div>
                </div>
            </div>

            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">
```

```
                    </div>
                </div>
            </div>

            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">

                    </div>
                </div>
            </div>

            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">

                    </div>
                </div>
            </div>

            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">

                    </div>
                </div>
            </div>

            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">

                    </div>
                </div>
            </div>
```

```
            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">

                    </div>
                </div>
            </div>

            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-header" id="card-head">
                        <b>Test</b>
                    </div>
                    <div class="card-body" id="card-body">

                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- jQuery first, then Tether, then Bootstrap JS. -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"

integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGF
F93hXpG5KkN" crossorigin="anonymous">
    </script>

    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper
.min.js"

integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPsk
vXusvfa0b4Q" crossorigin="anonymous">
    </script>

    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.j
s"

integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUa
r5+76PVCmYl" crossorigin="anonymous">
    </script>
</body>
```

```
</html>
```

Dodge.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" type="text/css" href="style.css">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/boots
trap.min.css"

integrity="sha384-rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoF
VhFQhNUwEyJ" crossorigin="anonymous">

    <script src="Functions.js"></script>
    <script src="dodge.js"></script>
    <script src="p5.js"></script>

    <title>Dodge</title>

</head>

<!-- Overskrift og home button -->

<body class="body">
    <div class="container-fluid">
        <div class="row">
            <div class="col order-first">
                <a
href="https://bigchungus420.github.io/EduGames/src/index.html"
class="image" title="Home Page"
                    style="position:absolute; left:25px; top:25px;">
                    <img src="images\home-button.png"
style="width:100px;height:100px">
                </a>
            </div>
            <div class="col">
                <div style="text-align: center;">
```

```
                    <h1 class="display-1" style="color:
aliceblue;"><b>Dodge</b></h1>
                </div>
            </div>
            <div class="col order-last"></div>
        </div>
    </div>

    <!-- Selve spil vinduet -->
    <div class="container-fluid">
        <div class="row justify-content-center" id="space-between">
            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-body" id="card-game">
                        <div id="canvas-holder"></div>
                    </div>
                </div>
            </div>
        </div>

        <!-- indstillinger og spil guide -->
        <div class="row justify-content-center" id="space-between">
            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-body" id="card-game-settings">
                        <div class="row justify-content-between m-4
mt-0">

                            <!-- Knapper til start, pause og genoptag
-->
                            <div class="col-auto">
                                <button type="button" class="btn
btn-success" id="startGame" value="Submit"> Start </button>
                            </div>
                            <div class="col-auto">
                                <button type="button" class="btn
btn-danger" id="pauseGame" value="Submit"> Pause </button>
                            </div>
                            <div class="col-auto">
                                <button type="button" class="btn
btn-primary" id="resumeGame" value="Submit"> Genoptag </button>
                            </div>
                        </div>
                        <div class="row justify-content-center">
                            <h3 style="color: aliceblue;
"><b>Sværhedsgrad</b></h3>
                        </div>
                        <div class="row">
```

```
                            <form>
                                <div class="form-group text-left">

                                    <!-- Slider som ændrer sværhedsgrad
fra 0 til 10. Startværdien er på 5 -->
                                    <input type="range" class="slider"
min="0" max="10" value="5" id="difficultySlider"
                                        style="background-color:
rgb(83, 21, 22); position: absolute; left: 20px">
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
            <!-- Her laves forklaringen til spillet "Styring" -->
            <div class="col-md-auto">
                <div class="card" id="card-stuff">
                    <div class="card-body" id="card-game-settings"
style="position: relative; top: 10px">
                        <div class="row justify-content-center">
                            <h2 style="color: aliceblue; "><b>
Styring</b></h2>
                        </div>
                        <div class="row">
                            <div class="col">
                                <ul style="color: aliceblue; font-size:
20px;">
                                    <li>(W A S D) Styrer op, venstre,
ned og højre</li>
                                    <li>Spillet går ud på at undgå
projektiler</li>
                                    <li>Hvis man bliver ramt, skal man
svare rigtigt på et spørgsmål for ikke at miste liv</li>
                                </ul>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- jQuery first, then Tether, then Bootstrap JS. -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"

integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGF
F93hXpG5KkN" crossorigin="anonymous">
```

```
    </script>

    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper
.min.js"

integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPsk
vXusvfa0b4Q" crossorigin="anonymous">
    </script>

    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.j
s"

integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUa
r5+76PVCmYl" crossorigin="anonymous">
    </script>


</body>

</html>
```

Litteraturliste

“Teknisk Matematik 4”, Preben Madsen
“Programmering”, Jesper Buch