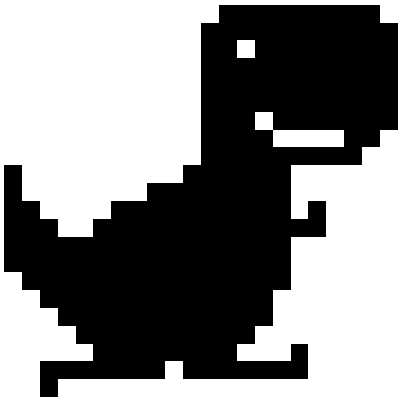
Programmering C

EUC Sjælland – HTX Næstved

Dino hop



Kris Kruse	Klasse: 3.A 2018/19

Udleveret:2019-02-18	Afleveret: 2019-05-13

Vejleder: Brian

Antal sider: 24 (m. bilag)

Indholdsfortegnelse

Abstract

Dette program er en klon af det kendte dinosaurspil der er indbygget i Googl Chrome. Det originale spil i Chrome vises hvis brugeren ikke har forbindelse til internettet når browseren er åben og startes med et tryk på enten mellemrum, 'W', 'S' eller piletasterne.

Denne version af spillet lavet i Processing, styres med 'W' 'S' og mellemrum samt udstyret med et gem system så du altid har dine highscores med når du spiller.

Projektbeskrivelse

Projektplan

Lave et spil i 2D af genren platformer/sidescroller, med inspiration fra googles dinohop (spillet der er i Chrome når man ikke har nogen internetforbindelse).

Der vil være fokus på de elementer der definere et spil:

- Nem menu med start spil funktion
- Score system
- Highscore liste
- Bruger styring

Krav

Jeg har valgt at programmere spillet i Processing, hvilket er et Java baseret programmeringssprog med ”integrated development enviroment (ide)” og indbygget grafisk bibliotek der gør det nemt at tegne med. (Processing.org, 2019) (Wikipedia, 2019)

Krav:

- Styre en sprite med piletasterne og eller ’w’ ’a’ ’s’ ’d’
- Animeret baggrund
- Score
- Highscore liste
- Menu
- Stigende sværhedsgrad

Udvide krav, der udføres hvis der bliver tid:

- Flere end en platform
- Pæne sprites/modeller
- Tilfældigt genereret bane

Testbeskrivelse

Følgende skal virke, når minimumskravene er opfyldt:

- Spiller spriten hopper som i virkeligheden
- Banen ændrer sig efterhånden som man når gennem den ´

Funktionalitet

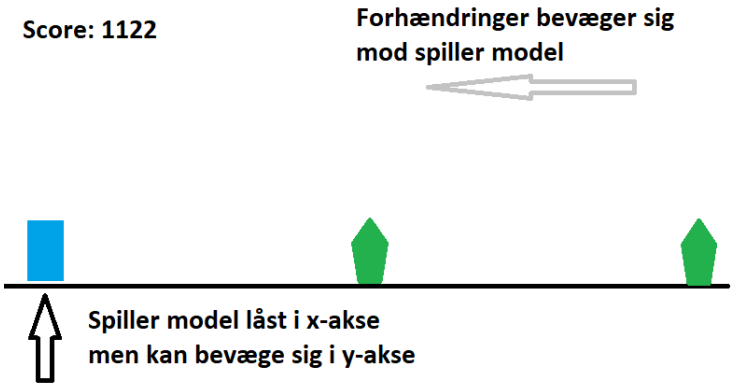
Skærmlayout

Forhindringer (grønne på figur 1), bevæger sig mod spilleren (blå). Scoren vises i top venstre.

Animation

Ikke vist i figur 1. Spiller figur skal have en simple løbe animation for at skabe en illusion af at spilleren bevæger sig. Derudover er der en animation når spilleren hopper og dukker sig.

Ekstra vil en af forhindringerne være en fugl der flyver



Figur 1 Skitse af skærmlayout

Funktionalitet i input

'g' starter spillet, 'mellemrum' fungerer som "hoppe" knap sammen med 'w', 's' fungerer som dukke knap.

Dokumentation

Udvikling af programmet

Med inspiration fra youtuberen Code bullet og hans "AI learns to play Chrome Dinosaur game || Can you beat it??" video. Link: https://youtu.be/sB_IGstiWlc

Da Code Bullet har indlagt et neuralt netværk som styrer af programmet skal rigtigt meget laves om og ekstra ting skal tilføjes for at give en beder bruger oplevelse.

Som start er der udarbejdet en base hvor nogle simple hoppefunktioner er sat op.

Da hoppe funktionen fungerede som forventet blev der implementeret forhindringer. Forhindringerne bevæger sig mod spilleren og skaber en illusion om at spilleren løber fremad. Da basen for forhindringerne var lavet, blev der opsat nogle basale hitboxe for spiller og forhindringer, samt lavet nogle fine sprites der passer til themet.

Figur 2. hopper.

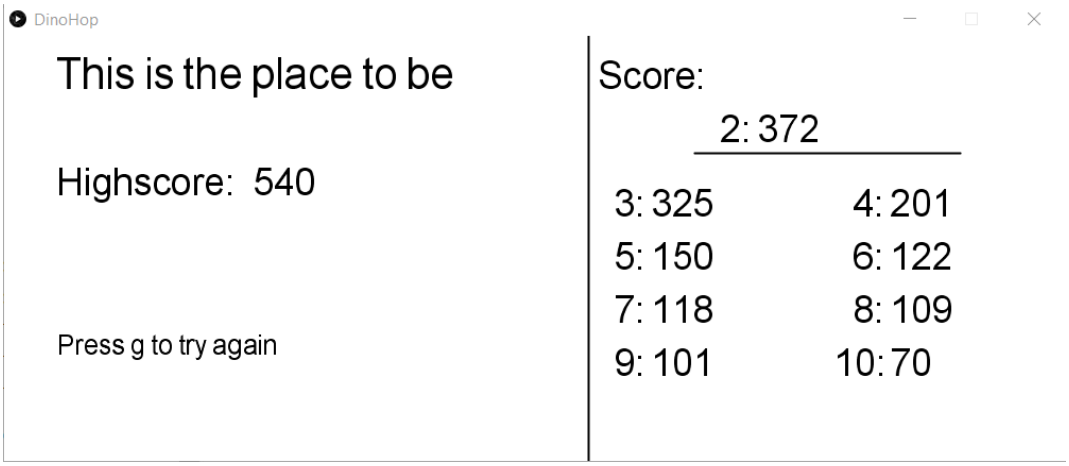
Da kollision med forhindringer er opsat laves en reset funktion der bliver kaldt når spilleren rammer en forhindring.

Da basissystemet fungerer og det begyndte at ligne et spil blev der fokuseret mere på bruger oplevelsen.

Som første prioritet var der i kravene en score og highscore system. Dette blev implementeret ved at kigge på mængden af tegnede frames fra spillet startede til spilleren ramte en forhindring. Men da denne værdi er normalt meget stor så den er skaleret en lille smule for beder at passe til spillet størrelse. Da der nu er blevet lagt en base for et scoresystem, kan den højeste score nemt blive logget og vist.

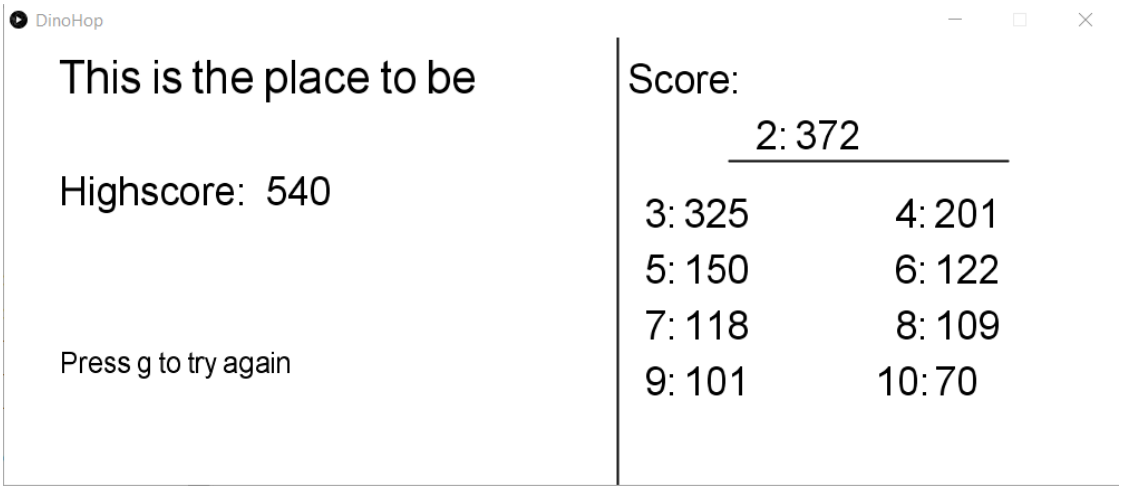
Da der nu er samlet nogle scores fra spilleren der blev gemt til et array, kan dette array skrives til en fil som et slags gem system. På denne måde er det muligt at gemme scoren fra spillet til næste gang spillet bliver åbent.

Som det sidste er udarbejdet en menu der kan vise spilleren de 10 bedste highscores, samt med muligheden for at starte et nyt spil. Denne menu bliver vist som den første skærm når spilleren åbner spillet op samt som game over skærm.

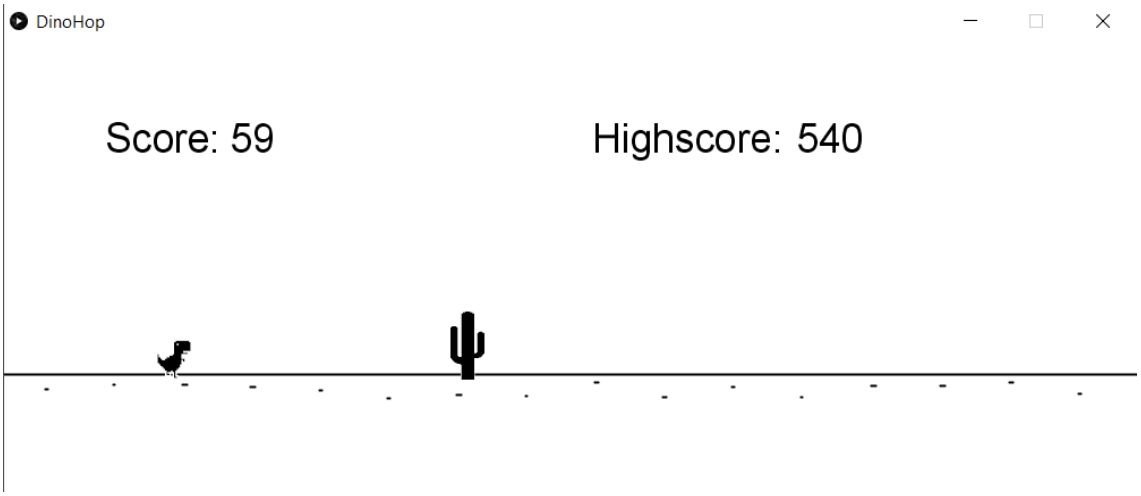


Figur 3. Game over og start skærm (Menu).

Skærmbilleder



Figur 4. Menu

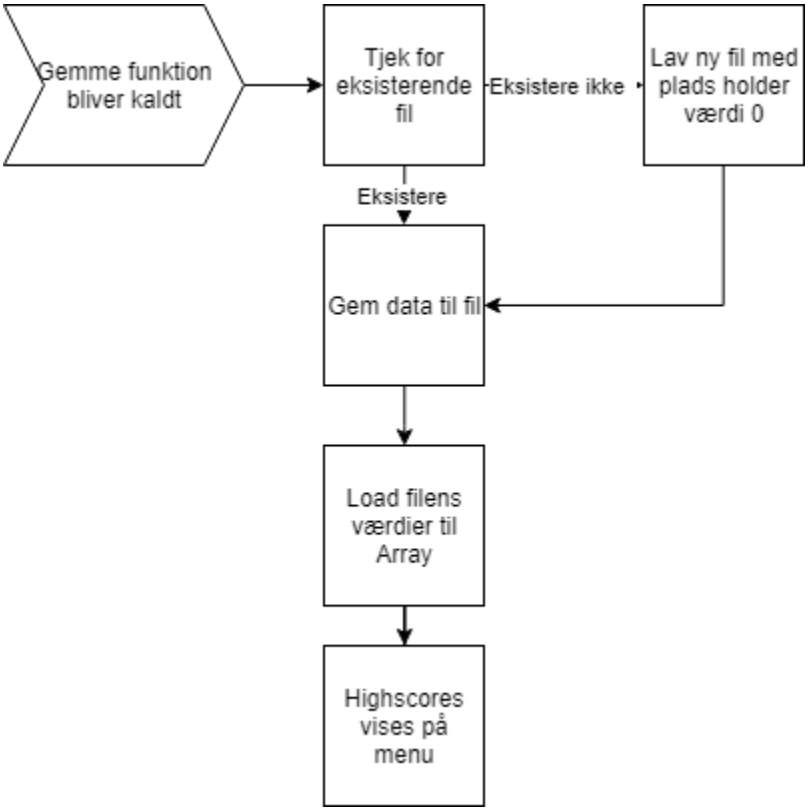


Figur 5. Aktivt spil

Udformningen af gemme funktion

Gemme funktionen virker på en sådan måde: Funktionen bliver kaldt, først tjekkes om der allerede eksistere en fil, hvis der ikke eksistere en fil laves en midlertidig fil. Når filen er blevet lavet eller hvis filen blev fundet tidligere, gemmes den nuværende data til filen hvorefter dataene indlæses til programmet igen og vises som Highscores. Der bliver indlæst igen for at sikre at dataene i fil og programmet stemmer overens.

Der bliver tjekket for om filen eksistere da et forsøg på at gemme i og eller indlæse fra en fil der ikke eksistere giver en nullexception fejl.



Figur 6. Flowchart over gemme funktion.

Selve koden har en fil og class for sig selv kaldet ”SaveSys” forkortelse af Save System.

Koden består af 3 funktioner; CheckForFile(), der tjekker om filen eksistere og laver en midlertidig hvis der ikke findes nogen. LoadFile(), der indlæser den gamle spil data fra filen. SaveFile() der gemmer den nye score til fil hvis den er højere end en af de 10 scores i filen.

Data fra spillet eller fra filen ved henholdsvis LoadFile() eller SaveFile() bliver lagt i arrayet x[]. Når der gemmes bliver dataet lavet om til en string for at kunne bruge processings indbyggede saveStrings() funktion. Data fra integer x[] arrayet bliver lagt om til string i[] arrayet.

```
1. class SaveSys{
2.   int HighScore = 0;
3.   SaveSys(){
4.   }
5.
6.
7.   void CheckForFile(){
8.    //Tjek om gem fil eksisterer, hvis ikke lav en dummy fil
9.     String path = dataPath("highscore.txt");
10.      File f = new File(path);
11.      if (f.exists() == false){
12.        //println("no exist");
13.        int[] x = {0,0,0,0,0,0,0,0,0,0};
14.        String[] i = str(x);
15.        saveStrings("/data/highscore.txt", i);
```

```
16.       }
17.     }
18.
19.     void LoadFile(){
20.       //Load highscore array fra fil og sorter, sotere for at være sikker
   på rækkefølge
21.       String[] load = loadStrings("data/highscore.txt");
22.       int[] x = int(load);
23.       x = sort(x);  //sætter x[0] som den mindste og resten i rækkefølge
   derefter
24.       HighScore = x[9];
25.       //println("done");
26.     }
27.
28.
29.     void SaveFile(int Score){
30.       //Load highscore array fra fil og sorter, sotere for at være sikker
   på rækkefølge
31.       String[] load = loadStrings("data/highscore.txt");
32.       int[] x = int(load);
33.       x = sort(x);  //sætter x[0] som den mindste og resten i rækkefølge
   derefter
34.
35.       //Overskriver den mindste highscore værdi med den nye og sortere
   værdierne igen
36.       if (Score > x[0]){
37.         x[0] = Score;
38.         x = sort(x);
39.         //println(x);
40.
41.         //Convetere array af int til et array af string og gemmer dataen
   til fil
42.         String[] savethis = str(x);
43.         saveStrings("/data/highscore.txt", savethis);
44.         HighScore = x[9];
45.
46.
47.
48.
49.       }
50.     }
51.   }
```

Følgende ting er testet med gemme systemet

Test	Forventet resultat	Resultat
Gem string til fil (fil eksisterer)	Ingen fejl, data gemt	Ingen fejl, data gemt
Gem string til fil (fil eksisterer ikke)	Ingen fejl, data gemt til ny lavet fil	Ingen fejl, data gemt til ny lavet fil
Indlæs fra fil (fil eksisterer)	Ingen fejl, data indlæst	Ingen fejl, data indlæst
Indlæs fra fil (fil eksisterer ikke)	Ingen fejl, data indlæst dog fra midlertidig 0 fil	Ingen fejl, data indlæst dog fra midlertidig 0 fil

Indlæs fra fil (fil ændret med notepad)	Ingen fejl, indlæser det der er i filen	Fejl; hvis dataet i filen blev ændret fra liste med 10, til liste med 9 skete en nullexceptionerror. Indlæser ellers ændrede værdier uden fejl.
Gem string til fil (fil ændret med notepad)	Ingen fejl, over skriver det der er i filen	Ingen fejl, over skriver det der er i filen

Forklaring på nullexceptionerror.

Den indlæste data kan godt gemmes til array, dog bliver der i flere tilfælde refereret til arrayets værdi i plads nummer 10 (x[9]), fejlen sker hvis denne plads er tom, altså har en 'null' værdi. Fejlen sker da programmet ikke kan regne med eller vise en tom værdi. Denne fejl kan nemt rettes ved at lave et check for array listens længde inden den sidste værdi aflæses og vises.

Brugervejledning

Når spillet er startet og hovedmenuen er åben kan spillet startes med et tryk på 'G'.

Spillet er styret med 'W' og mellemrum for hop samt 'S' for at dukke sig.

Nuværende score kan ses lige over spilleren på spille skærmen, nuværende højeste score kan ses til højre for denne.

De 10 bedste scores kan ses på hovedmenuskærmen.

Test

Der er løbende foretaget tests af programmets funktioner, undervejs.

Den endelige egen test

Bruger	Program
Tryk på 'W' eller mellemrum	I spil: Dino hopper I menu: ikke noget
Tryk på 'S'	I spil: Dino falder hurtigt hvis i luften, ellers dukker den sig I menu: ikke noget
Tryk på 'G'	I spil: ikke noget I menu: spil starter

Konklusion og perspektivering

Spillet virker som det skal med bruger inputs og køre godt.

Kravene opskrevet i projektets begyndelse er mere eller mindre klaret. Dette kan ses da, spriten ”Dino” er bruger styret med ’W’ ’S’ og mellemrum. Baggrunden er som sådan ikke animeret men forgrunden/jorden er. Score system er implementeret og fungere. Sammen med denne er lavet en highscore liste med de 10 bedste scores der alle bliver gemt til fil. Der er implementeret en menu der vises til brugeren ved start og mellem spil. Spillet har en stigende sværhedsgrad efterhånden som den overlevede tid stiger.

Derudover er der implementeret et gemme system så de 10 bedste scores bliver gemt til næste gang spillet åbnes, denne er dog ikke fejlfri men kan hurtigt fikses hvis der implementeres et tjek for manglende pladser i gemme arrayet.

Referencer

Processing.org. (11. Maj 2019). *Processing*. Hentet fra Processing.org: https://processing.org/

Wikipedia. (11. Maj 2019). *Processing (programming language): Wikipedia*. Hentet fra Wikipedia: https://en.wikipedia.org/wiki/Processing_(programming_language)

Bilag

Bilag 1: Logbog

Mandag den 18-02-2019

Forsøg på valg af emne, tænker et spil noget af samme stil som Chromes ingen internet dinosaur. Valg af programmeringssprog mangler.

Mandag den 25-02-2019

Valgt programmeringssprog Processing, en side gren af Java, til at skrive projektet i.

Projektbeskrivelse lavet

Tirsdag den 26-02-2019

Projektbeskrivelse godkendt, arbejde er begyndt. Har i dag forsøgt at lære syntax for Processing samt prøve at tegne et basalt canvas for at finde ud af hvordan programmeringssproget virker.

Søndag den 03-03-2019

Projektet er blevet lagt op på Github så jeg har nem adgang til at lave noget lige meget hvor jeg er. Tilføjet textures til spiller sprite og obstacles. Fikset Dinoens hop, så nu er der basis bevægelse for spilleren der aktiveres med 'space' knappen. Dino kan nu også dukke sig, dog er denne funktion stadig lidt underlig.

Mandag den 11-03-2019

Implementeret hitbokse på forhindringerne samt implementeret en reset funktion i tilfælde af at spilleren rammer en forhindring. Forsøgt at implementere en højere hoppe funktion hvis en af hoppe knapperne bliver holdt nede.

Mandag den 18-03-2019

Implementeret en nuværende score viser, så det skulle ikke være så svært at få lavet en highscore viser også, samt en måde at gemme highscoren til næstegang spillet åbnes. Den højere hoppe funktion fungerede ikke ordentligt og er pt blevet kommenteret ud mens jeg finder en måde at lave den på ordentligt. Ekstra kommentarer er tilføjet for at hjælpe til forståelsen, mangler stadig en del af koden, samt strukturer den lidt beder.

Mandag den 01-04-2019

Highscore system implementeret, den højeste highscore bliver vist mens de 10 bedste bliver gemt til en fil i data mappen. Highscore filen bliver opdateret hvis den laveste score bliver slået. 1

Tirsdag den 16-04-2019

Implementeret en start menu der kommer frem umiddelbart efter at man har tabt spillet samt når spillet startes op.

Bilag 2: Programudskrift

Fil DinoHop.dpe

```
1. PFont font;
2. Player player;
3. SaveSys savesys = new SaveSys();
4. Menu Menu = new Menu();
5.
6. //Billeder
7. PImage dinoRun1;
8. PImage dinoRun2;
9. PImage dinoJump;
10.  PImage dinoDuck;
11.  PImage dinoDuck1;
12.  PImage smallCactus;
13.  PImage manySmallCactus;
14.  PImage bigCactus;
15.  PImage bird;
16.  PImage bird1;
17.
18.
19.  //---------------------Variable-----------------------------------
20.
21.  int groundHeight = 100;  //Jord linjens højde fra bunden af frame
22.  int playerXpos = 150;    //spillers placering fra venstre væg
23.  int score = 0;           //Score spilleren har opnået
24.
25.  int obstacleTimer = 0;
26.  int minimumTimeBetweenObstacles = 60;
27.  int randomAddition = 0;
28.  int groundCounter = 0;
29.  float speed = 10;
30.
31.  float Birdcount = 0.15;  //bestemmer hvor mange procent er fugle. 1 =
   100%
32.
33.
34.  ArrayList<Obstacle> obstacles = new ArrayList<Obstacle>();
35.  ArrayList<Bird> birds = new ArrayList<Bird>();
36.  ArrayList<Ground> grounds = new ArrayList<Ground>();
37.
38.  Boolean noFly = false;
39.
40.  Boolean menu = true;
41.
42.  //------------Variable slut-------------------------------------
43.
44.
45.  //------------Setup, Køre en gang-------------------------------
46.
47.  void setup(){
48.    //----canvas indstillinger
49.    frameRate(60);
50.    size(1000,400);
51.
52.    //-----DATA load test
53.    savesys.CheckForFile();
54.    savesys.LoadFile();  //Loader highscoren fra sidst
55.
```

```
56.
57.
58.
59.    //------Textures
60.    dinoRun1 = loadImage("dinorun0000.png");
61.    dinoRun2 = loadImage("dinorun0001.png");
62.    dinoJump = loadImage("dinoJump0000.png");
63.    dinoDuck = loadImage("dinoduck0000.png");
64.    dinoDuck1 = loadImage("dinoduck0001.png");
65.
66.    smallCactus = loadImage("cactusSmall0000.png");
67.    bigCactus = loadImage("cactusBig0000.png");
68.    manySmallCactus = loadImage("cactusSmallMany0000.png");
69.    bird = loadImage("berd.png");
70.    bird1 = loadImage("berd2.png");
71.
72.
73.
74.    //------font og andet
75.    font = loadFont("Arial.vlw");
76.    textFont(font,32);
76.    player = new Player();
78.
79.  }
80.
81.  //------------Setup, slut----------------------------------------
82.
83.
84.  //------------Draw, køre hele tiden-----------------------------
85.
86.  void draw(){
87.    //Hvis menuen er sat til at være aktiv køres menu koden
88.    if (menu) {
89.      Menu.MenuSide();
90.
91.     //Spil kørsels koden
92.    }else{
93.
94.    score = frameCount/2;
95.    smooth();
96.    drawToScreen();
97.    player.show();
98.    player.move();
99.    updateObstacles();
100.
101.   //text(obstacles.size(),300,100);
102.   //text(birds.size(),400,100);
103.   text("Score: ",90,100);
104.   text(score,200,100);
105.   text("Highscore:",520,100);
106.   text(savesys.HighScore, 700,100);
107.
108.   fill(0);
109.
110.
111.
112.   player.update();
113.
114.   }
115. }
116.
```

```
117. //------------Draw, slut----------------------------------------
118.
119. //------------Input---------------------------------------------
120. void keyPressed(){
121.
122.   switch(key){
123.     case ' ':          //Funktion for tryk på 'Space'/'mellemrum' key på
   keyboard
124.     player.jump(false);
125.       break;
126.
127.      case 'w':        //Funktion for tryk på w key på keyboard
128.      player.jump(false);
129.       break;
130.
131.      case 'g':        //lukker for menuen
132.      if (menu){
133.        menu = false;
134.        frameCount = 0;
135.        score = 0;
136.
137.      }
138.
139.      break;
140.
141.     case 'b':         //Funktion for tryk på b key på keyboard
142.       exit();
143.       break;
144.   }
145.   if (key == 's'){    //Funktion for tryk på s key på keyboard
146.     player.ducking(true);
147.   }else {
148.   player.ducking(false);
149.   }
150. }
151. //------------Input slut----------------------------------------
152.
153. //------------tegn på skærm-------------------------------------
154.
155. void drawToScreen(){
156.   background(255);      //Baggrund overskrevet til hvid
157.   stroke(0);
158.   strokeWeight(2);
159.   line(0, height - groundHeight - 5, width, height - groundHeight - 5);
160.
161.
162. }
163.
164. //------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ---------
165. //opdatere forhændringerne hvert frame
166. void updateObstacles() {
167.   obstacleTimer ++;
168.   speed += 0.002;
169.   if (obstacleTimer > minimumTimeBetweenObstacles + randomAddition) {
   //Hvis obstacle timer er høj nok tilføj en ny obstacle
170.     addObstacle();
171.   }
172.   groundCounter ++;
173.   if (groundCounter> 5) { //Hvert 10 frames tilføj en ground bit
```

```
174.     groundCounter =0;
175.     grounds.add(new Ground());
176.   }
177.
178.   moveObstacles();    //Flyt alting til venstre
179.   showObstacles();
180. }
181.
182.
183.
184. //------------------------------------------------------------------------
     ------------------------------------------------------------------------
     ---------
185. //Bevæg alt til venstre baseret på hastigheden af spillet
186. void moveObstacles() {
187.   //println(speed);
188.   for (int i = 0; i< obstacles.size(); i++) {
189.     obstacles.get(i).move(speed);
190.     if (obstacles.get(i).posX < -playerXpos) {
191.       obstacles.remove(i);
192.       i--;
193.     }
194.   }
195.
196.   for (int i = 0; i< birds.size(); i++) {
197.     birds.get(i).move(speed);
198.     if (birds.get(i).posX < -playerXpos) {
199.       birds.remove(i);
200.       i--;
201.     }
202.   }
203.   for (int i = 0; i < grounds.size(); i++) {
204.     grounds.get(i).move(speed);
205.     if (grounds.get(i).posX < -playerXpos) {
206.       grounds.remove(i);
207.       i--;
208.     }
209.   }
210. }
211.
212. //------------------------------------------------------------------------
     ------------------------------------------------------------------------
     ------------
213. //Tilføjer en forhændring til banen
214. void addObstacle() {
215.   int lifespan = score;
216.   int tempInt;
217.   if (lifespan > 1000 && random(1) < Birdcount) {
218.     tempInt = floor(random(3));
219.     Bird temp = new Bird(tempInt);//floor(random(3)));
220.     birds.add(temp);
221.   } else {//otherwise add a cactus
222.     tempInt = floor(random(3));
223.     Obstacle temp = new Obstacle(tempInt);//floor(random(3)));
224.     obstacles.add(temp);
225.     tempInt+=3;
226.   }
227.
228.   randomAddition = floor(random(50));
229.   obstacleTimer = 0;
230. }
```

```
231.
232. //------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ---------
233. //Hviser Fohændringer
234. void showObstacles() {
235.   for (int i = 0; i< grounds.size(); i++) {
236.     grounds.get(i).show();
237.   }
238.   for (int i = 0; i< obstacles.size(); i++) {
239.     obstacles.get(i).show();
240.   }
241.
242.   for (int i = 0; i< birds.size(); i++) {
243.     birds.get(i).show();
244.   }
245. }
246.
247.
248. //------------------------------------------------------------------------
   ------------------------------------------------------------------------
249. //Resetter banen og score systemet når dinoen rammer en forhændring
250. void resetObstacles() {
251.   obstacles = new ArrayList<Obstacle>();
252.   birds = new ArrayList<Bird>();
253.   obstacleTimer = 0;
254.   randomAddition = 0;
255.   groundCounter = 0;
256.   speed = 10;
257.   savesys.SaveFile(score);
258.   frameCount = 0;
259.   score = 0;
260.   menu = true;
261. }
```

Fil Birds.pde

```
1. class Bird {
2.   float w = 60;
3.   float h = 50;
4.   float posX;
5.   float posY;
6.   int flapCount = 0;
7.   int typeOfBird;
8. //------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ----
9.  //constructor
10.   Bird(int type) {
11.     posX = width;
12.     typeOfBird = type;
13.     switch(type) {
14.     case 0://flying low
15.       posY = 10 + h/2;
16.       break;
17.     case 1://flying middle
18.       posY = h;
19.       break;
20.     case 2://flying high
21.       posY = h*2;
```

```
22.         break;
23.       }
24.     }
25.   //------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ------
26.     //show the birf
27.     void show() {
28.       flapCount++;
29.
30.       if (flapCount < 0) {//flap the berd
31.         image(bird,posX-bird.width/2,height - groundHeight - (posY +
   bird.height-20));
32.       } else {
33.         image(bird1,posX-bird1.width/2,height - groundHeight - (posY +
   bird1.height-20));
34.       }
35.       if(flapCount > 15){
36.        flapCount = -15;
37.
38.       }
39.     }
40.   //------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ------
41.     //move the bard
42.     void move(float speed) {
43.       posX -= speed;
44.     }
45.   //------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ------
46.     //returns whether or not the bird collides with the player
47.     boolean collided(float playerX, float playerY, float playerWidth,
   float playerHeight) {
48.
49.       float playerLeft = playerX - playerWidth/2;
50.       float playerRight = playerX + playerWidth/2;
51.       float thisLeft = posX - w/2 ;
52.       float thisRight = posX + w/2;
53.
54.       if ((playerLeft<= thisRight && playerRight >= thisLeft ) ||
   (thisLeft <= playerRight && thisRight >= playerLeft)) {
55.         float playerUp = playerY + playerHeight/2;
56.         float playerDown = playerY - playerHeight/2;
57.         float thisUp = posY + h/2;
58.         float thisDown = posY - h/2;
59.         if (playerDown <= thisUp && playerUp >= thisDown) {
60.           return true;
61.         }
62.       }
63.       return false;
64.     }
65.   }
```

Fil Ground.pde

```
1. //Viser små prikker på jorden
2. class Ground{
3.   float posX = width;
4.   float posY = height -floor(random(groundHeight - 20,groundHeight));
5.   int w = floor(random(1,5));
6.
7.   Ground(){}
8.
9.   void show(){
10.      stroke(0);
11.      strokeWeight(2);
12.      line(posX,posY, posX + w, posY);
13.
14.    }
15.    void move(float speed) {
16.      posX -= speed;
17.    }
18.  }
```

Fil Menu.pde

```
1. class Menu{
2.   SaveSys savesys = new SaveSys();
3.   Menu(){}
4.
5. void MenuSide(){
6.
7.   //Load highscore array fra fil og sorter, sotere for at være sikker på
   rækkefølge
8.   String[] load = loadStrings("data/highscore.txt");
9.   int[] x = int(load);
10.    x = sort(x);  //sætter x[0] som den mindste og resten i rækkefølge
   derefter
11.    //println("done");
12.
13.
14.
15.    smooth();
16.    background(255);      //Baggrund overskrevet til hvid
17.    stroke(0);
18.    strokeWeight(2);
19.
20.    //Venstre side af canvas
21.    textSize(40);
22.    text("This is the place to be",50,50);
23.    textSize(35);
24.    text("Highscore:",50,150);
25.    text(x[9],235,150);
26.    textSize(25);
27.    text("Press g to try again",50,300);
28.
29.
30.
31.
32.    //SCORES på højre side af canvas
33.    textSize(35);
34.    line(550, 0, 550, height);
35.    text("Score:",560,50);
36.
37.
```

```
38.    text("2:",675,100);
39.    text(x[8],710,100);
40.    line(650,110,900,110);
41.
42.    //Venstre side har alle ulige tal
43.    text("3:",575,170);
44.    text(x[7],610,170);
45.
46.    text("5:",575,220);
47.    text(x[5],610,220);
48.
49.    text("7:",575,270);
50.    text(x[3],610,270);
51.
52.    text("9:",575,320);
53.    text(x[1],610,320);
54.
55.    //Højre side har lige tal
56.    text("4:",800,170);
57.    text(x[6],835,170);
58.
59.    text("6:",800,220);
60.    text(x[4],835,220);
61.
62.    text("8:",800,270);
63.    text(x[2],835,270);
64.
65.    text("10:",782,320);
66.    text(x[0],835,320);
67.
68.
69.    fill(0);
70.
71.
72.    }
73.  }
```

Fil Obstacle.pde

```
1. class Obstacle {
2.   float posX;
3.   int w ;
4.   int h ;
5.   int type;
6.   //------------------------------------------------------------------------
   ------------------------------------------------------------------------
   ------
7.   //constructor
8.   Obstacle(int t) {
9.     posX = width;
10.      type = t;
11.      switch(type) {
12.      case 0://small cactus
13.        w = 40;
14.        h = 40;
15.        break;
16.      case 1://big cactus
17.        w = 60;
18.        h = 60;
19.        break;
```

```
20.       case 2://small cacti
21.         w = 100;
22.         h = 40;
23.         break;
24.       }
25.     }
26.
27.     //------------------------------------------------------------------------
    ------------------------------------------------------------------------
    --------
28.     //show the cactus
29.     void show() {
30.       fill(0);
31.       rectMode(CENTER);
32.       switch(type) {
33.       case 0:
34.         image(smallCactus, posX - smallCactus.width/2, height -
    groundHeight - smallCactus.height);
35.         break;
36.       case 1:
37.         image(bigCactus, posX - bigCactus.width/2, height - groundHeight -
    bigCactus.height);
38.         break;
39.       case 2:
40.         image(manySmallCactus, posX - manySmallCactus.width/2, height -
    groundHeight - manySmallCactus.height);
41.         break;
42.       }
43.     }
44.     //------------------------------------------------------------------------
    ------------------------------------------------------------------------
    --------
45.     // move the obstacle
46.     void move(float speed) {
47.       posX -= speed;
48.     }
49.     //------------------------------------------------------------------------
    ------------------------------------------------------------------------
    --------
50.     //returns whether or not the player collides with this obstacle
51.     boolean collided(float playerX, float playerY, float playerWidth,
    float playerHeight) {
52.
53.       float playerLeft = playerX - playerWidth/2;
54.       float playerRight = playerX + playerWidth/2;
55.       float thisLeft = posX - w/2 ;
56.       float thisRight = posX + w/2;
57.
58.       if ((playerLeft <= thisRight && playerRight >= thisLeft ) ||
    (thisLeft <= playerRight && thisRight >= playerLeft)) {
59.         float playerDown = playerY - playerHeight/2;
60.         float thisUp = h;
61.         if (playerDown <= thisUp) {
62.           return true;
63.         }
64.       }
65.       return false;
66.     }
67.   }
```

Fil Player.dpe

```
1. class Player {
2.   float posY = 0;
3.   float velY = 0;
4.   float gravity = 0.5;
5.
6.   Boolean dead = false;
7.   Boolean duck = false;
8.   int runCount = -5;
9.
10.    int size = 20;
11.    Player(){
12.    }
13.
14.     void show() {
15.
16.       //----opdatere sprite, skaber en animation når dino hopper eller
   dukker---
17.      if (duck && posY == 0) {
18.        if (runCount < 0) {
19.
20.          image(dinoDuck, playerXpos - dinoDuck.width/2, height -
   groundHeight - (posY + dinoDuck.height));
21.        } else {
22.
23.          image(dinoDuck1, playerXpos - dinoDuck1.width/2, height -
   groundHeight - (posY + dinoDuck1.height));
24.        }
25.      } else
26.        if (posY ==0) {
27.          if (runCount < 0) {
28.            image(dinoRun1, playerXpos - dinoRun1.width/2, height -
   groundHeight - (posY + dinoRun1.height));
29.          } else {
30.            image(dinoRun2, playerXpos - dinoRun2.width/2, height -
   groundHeight - (posY + dinoRun2.height));
31.          }
32.        } else {
33.          image(dinoJump, playerXpos - dinoJump.width/2, height -
   groundHeight - (posY + dinoJump.height));
34.        }
35.     runCount++;
36.     if (runCount > 5){
37.       runCount = -5;
38.     }
39.    }
40.
41.   //----------Dinos bevægelser og tyngdekraftens påvirkning-------------
42.   void move(){
43.     posY += velY;
44.
45.     if (posY > 0){
46.       velY -= gravity;
47.
48.     }else{
49.       velY = 0;
50.       posY = 0;
51.     }
52.     if (keyPressed != true){
```

```
53.       duck = false;
54.     }
55.     //-----tjekker for obstacle hits
56.        for (int i = 0; i< obstacles.size(); i++) {
57.          if (obstacles.get(i).collided(playerXpos, posY +
   dinoRun1.height/2, dinoRun1.width*0.5, dinoRun1.height)) {
58.            resetObstacles();
59.          }
60.        }
61.
62.        for (int i = 0; i< birds.size(); i++) {
63.          if (duck && posY ==0) {
64.            if (birds.get(i).collided(playerXpos, posY +
   dinoDuck.height/2, dinoDuck.width*0.8, dinoDuck.height)) {
65.              resetObstacles();
66.            }
67.          }
68.        }
69.   }
70.   //----------bukker sig
71.     void ducking(boolean isDucking) {
72.      if (posY != 0 && isDucking) {
73.        gravity = 3;
74.      }else {
75.      gravity = 0.5;
76.      }
77.      duck = isDucking;
78.    }
79.
80.   //---------hopper
81.     void jump(boolean bigJump) {
82.      if (player.posY == 0){
83.      gravity = 1;
84.      velY = 9;
85.      }
86.       //---------forsøg på højt hop
87.      // if (player.posY >= 0 && player.posY < 80 && noFly == false) {
88.      //  if (bigJump) {
89.      //    gravity = 1;
90.      //    velY = 15;
91.      //    noFly = true;
92.      //  } else {
93.      //    gravity = 1.2;
94.      //    velY = 9;
95.      //  }
96.      //}
97.    }
98.
99.
100.
101.  void update(){
102.
103.
104.  }
105.
106.
107.
108. }
```

Fil SaveSys.pde

```
1. class SaveSys{
2.   int HighScore = 0;
3.   SaveSys(){
4.   }
5.
6.
7.   void CheckForFile(){
8.    //Tjek om gem fil eksistere, hvis ikke lav en dummy fil
9.     String path = dataPath("highscore.txt");
10.      File f = new File(path);
11.      if (f.exists() == false){
12.        //println("no exist");
13.        int[] x = {0,0,0,0,0,0,0,0,0,0};
14.        String[] i = str(x);
15.        saveStrings("/data/highscore.txt", i);
16.      }
17.    }
18.
19.    void LoadFile(){
20.      //Load highscore array fra fil og sorter, sotere for at være sikker
   på rækkefølge
21.      String[] load = loadStrings("data/highscore.txt");
22.      int[] x = int(load);
23.      x = sort(x);  //sætter x[0] som den mindste og resten i rækkefølge
   derefter
24.      HighScore = x[9];
25.      //println("done");
26.    }
27.
28.
29.    void SaveFile(int Score){
30.      //Load highscore array fra fil og sorter, sotere for at være sikker
   på rækkefølge
31.      String[] load = loadStrings("data/highscore.txt");
32.      int[] x = int(load);
33.      x = sort(x);  //sætter x[0] som den mindste og resten i rækkefølge
   derefter
34.
35.      //Overskriver den mindste highscore værdi med den nye og sortere
   værdierne igen
36.      if (Score > x[0]){
37.        x[0] = Score;
38.        x = sort(x);
39.        //println(x);
40.
41.        //Convetere array af int til et array af string og gemmer dataen
   til fil
42.        String[] savethis = str(x);
43.        saveStrings("/data/highscore.txt", savethis);
44.        HighScore = x[9];
45.
46.
47.
48.
49.      }
50.    }
51.  }
```