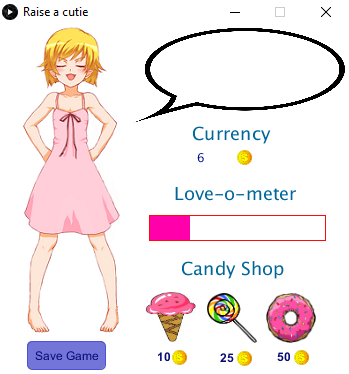
25-04-2016

Alan Slivany

2.R

Programmering C

Eksamensprojekt



Indholdsfortegnelse

[Abstrakt 2](#_Toc449311651)

[Problemformulering 2](#_Toc449311652)

[Funktionsbeskrivelse 2](#_Toc449311653)

[Layout 2](#_Toc449311654)

[Problemløsning 5](#_Toc449311655)

[Tidsplan 5](#_Toc449311656)

[Dokumentation af programmet 5](#_Toc449311657)

[Flowcharts 6](#_Toc449311658)

[Beskrivelse af kode 6](#_Toc449311659)

[Inddragelse af matematik 6](#_Toc449311660)

[Test af programmet 8](#_Toc449311661)

[Konklusion 9](#_Toc449311662)

[Bilag 9](#_Toc449311663)

# Abstrakt

Menneskeheden udvikler sig nu hurtigere end nogensinde før og det gælder også udviklingen af computer spil. Størstedelen af computer spil er lange og man skal ofte give spillet sit fulde koncentration, men behøves det at være sådan? Findes der spil, hvis sekundære funktion egentlig er at underholde forbrugeren, hvor dets primære funktion er noget helt andet?   
Dette udforskes i denne journal, hvor der tages et kig på et spil navngivet *Raise a cutie*, hvis mål ikke er hvad der vil ses i et klassiks spil.

# Problemformulering

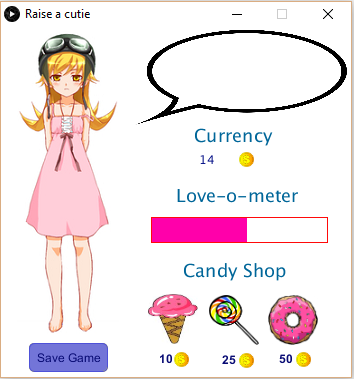
Alt for ofte bruger folk en computer over længere perioder og det kan for nogle være meget udmattende og demotiverende. Dette kan især fremstår hvis det man sidder og lavet på sin computer virker gentagende. Det er dog vigtigt at holde en pause og åbne sit sind. Det kan gøres på forskellige måder, blandet andet ved at stå op og gå en tur eller tjekke Facebook.

Dette er hvor spillet kommer ind, da spillet ikke er et normalt spil i almindelighedens forstand, men mere et sekundært spil man vil have kørende i baggrunden. Så efter udmattelse bruger man et par minutter på spillet og får samtidigt lidt øje guf så er man videre igen.

# Funktionsbeskrivelse

## Layout

Som vist i *billede* kommer spillet i et fin lille vindue, som ikke er mere end 350x350. Det første man bliver mødt med når spillet åbnes er en lille lyshåret pige og nogle andre brugerfladere og illustrationer. Disse andre brugerflader og illustrationer består af en taleboble, noget tekst, nogle penge, en meter, noget slik og en knap til at gemme spillet. Det er et simpelt og relativt minimalistisk design som bruger alt det plads der er til rådighed.   
I forhold til funktion er spillet også ret simpel og lige til. De gyldne guldmønter, såkaldte mønter, er optjent ved at klikke på billedet af hovedkarakteren. Et klik belønnes med en mønt og disse mønter kan så bruges ned i slikkebutikken, hvor man enten kan købe en is, slikkepind eller donut.  
Dette slik giver XP, og det vil sige at det illustreret meter vil stige hver gang der købes slik. Jo dyre slik der købes, desto mere XP bliver der giver og jo højere stiger vores meter i så fald.  
Udover det er der en blå knap under hovedkarakteren som kan gemme spillet. Når denne knap trykkes, gemmes en fremskridt og man ville kunne lukke og åbne spillet igen og fortsatte hvor man slap.



(Billede 1)

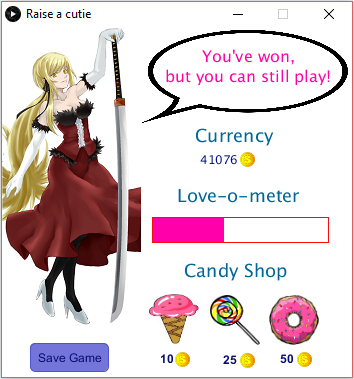
Der er i spillet 6 såkaldte ’levels’, altså niveauer, hvorved hovedkarakteren vokser en smule for hvert level hun stiger. Dette illustreres på *billede 2*, hvor der fra start er vist hovedkarakteren i level 1, hen op til sidst hvor hovedkarakteren er i level 6. Det kan tydeligt ses at der for hvert level fremkommer et nyt billede, hvor hovedpersonen ser lidt ældre ud. Udover denne ændring af billedet af hovedkarakteren, er der ikke andre ændringer på noget andet og hovedkarakter vil ikke få et nyt billede efter level 6. Efter level 6 vil man derfor have vundet spillet.



(Billede 2)

Men selvom spillet er vundet efter at have klaret level 6, er det stadig muligt at spille videre og det vil stadig blive sværere. Dette kan ses på *billede 3* hvor hovedkarakter siger at man har vundet , men at man stadig kan fortsætte. Heraf fortsætter man med at klikke og købe som man hen af vejen har gjort og sådan kan man forsætte i alt uendelighed - selvom det begynder at blive meget sværere jo længere man kommer op i levels.

(Billede 3)



# Problemløsning

Programmet er lavet i programmeringssproget Processing 3.

Følgende biblioteker er blevet brugt:

* G4P (GUI for Processing) af Peter Lager
* Let optimering af brugerflade.
* Let at filføje og bruge simple image buttons.
* Minim af Damien Di Fede og Anderson Mills
* Kan af udløse lyd ved bestemte funktioner.

# Tidsplan

# Dokumentation af programmet

## Flowcharts



(Flowchart 1)

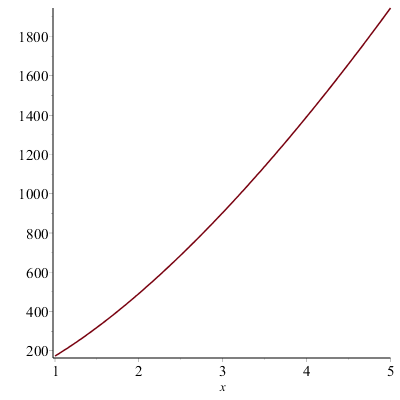
*Flowchart 1* viser en beregning af såkaldte ’love-o-meter’ og hvad der egentlig sker når der bliver givet XP, i forhold til billedeskift. Det første der sker efter start er at xpTillevel bliver regnet ud, og den bestemmer hvor meget XP der skal opnås per level. Den stiger så gradvist afhængig af hvad level man er i. Efter at have købt noget slik kan få noget af dette XP, dette er dog ikke illustreret her på *flowchart 1*. Når der optjenes noget XP, vil spillet tjekker om vores meter er større eller lig med den før beregnede xpTilLevel. Hvis den er det, så kalder vi op vores lvlUP funktion, som blandt andet nulstiller vores meter, leveler op og skifter billede. Herefter udregnes den nye xpTilLevel og som nævnt er den afhængig af nuværende level, så den vil være endnu højere denne gang. Hvis vores meter ikke er større eller lig med xpTilLevel så beholder biledes og der bliver gået tilbage til engang før experience increased.

## Beskrivelse af kode

### Inddragelse af matematik

|  |
| --- |
| public void **draw**() {    background(255);    //Potensfunktion    xpTilLevel = 174\*pow(cutie.lvl+1, 1.5);  } |

I kodningen er der gjort brug af matematik i form af procentregning og en potens funktion. Kigges der på 4. linje af koden, refereret til under potensfuntkion, ses den potensfunktion som der er blevet brugt. Pow er en metode som tillader at opløfte tal i Processing, da det ikke ordret kan skrives ned. Så det der essentielt står er , hvor x er hvilket level man på derværende tidspunkt er i. Dette er den protensfunktionen der bliver benyttet til at forhøje mængden af experience der skal opnås før et level, for hvert level man stiger.  
Der er nedenunder i *graf 3* illustreret denne potensfunktion fra level 1 til 6. Der kan tydelig ses at grafem stiger gevaldigt og at man skal bruge en del mere XP til at stige level når man er i level 6, i forhold til hvis man nu var i level 1.



(Graf 1)

|  |
| --- |
| public void **draw**() {    background(255);    //Filling love-o-meter    fill(255, 0, 170);    float barLength = floor((174/xpTilLevel)\*meter);    rect(151, 191, barLength, 24);    noFill();    noStroke();  } |

Ses der videre til den næste del af koden, hvor der bliver brugt matematik, finder vi vores kærlighedsbar, som indeholder noget procentregning. Det væsentlige at kigge op her er ved linje 7, lige under fill. Det er simpel procentregning der bruges til beregning af hvor langt inde i kærlighedsmeteren vi er. Men før der tales dybere ind til det, er det vigtigt der skabes en forståelse for hvad floor er for en metode. Metoden floor tillader afrunding af tal, men det er ikke helt almindelig afrunding. floor afrunder altid ned uafhængigt af hvilket decimal vi får ud. Det kunne være tallet inde i metoden gav 2,88, men det ville stadig blive rundet ned til 2. Grunden til at floor her bliver brugt er meget simpel. Det er fordi, som tidligere nævnt, at barLength er længden på kærlighedsbaren, altså hvor meget XP man har på nuværende tidspunkt. Og det er ikke muligt at vise f.eks. en halv pixel eller en halvanden pixel, derfor er afrunding nødvendigt.  
Med den information på plads, kan vi kigge ind i metoden for der sker noget procentregning. Her bliver bare gjort brug af den velkendte procentformel , hvor r er procent af et tal *K* som kan findes ved brug af formlen. Men det går stadig ikke helt op med den information der er givet på nuværende tidspunkt fordi vi ved ikke hvad meter er. Det skyldes at meter bliver brugt i andre classes og er derved svær at vise. Men kort fortalt er meter det variable der bliver tilføjet på hver gang der købes noget slik, som f.eks. en slikkepind (meter += icecream;).  
Så det der sker er at 174 bliver divideret med xpTilLevel, som holder hvor meget experience man skal have før man stiger leve, som der tidligere er blevet talt om. Herefter ganges det tal der kommer ud med meter, som der lige er blevet forklaret. Dette resulterer så i et tal, som er barLength og det er ikke et helt tilfældigt tal, tallet er nemlig spillerens fremskridt i procent.

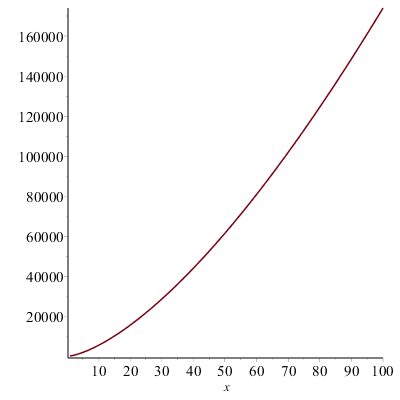
# Test af programmet

Her er på *billede 4* udøvet en simpel lille test, hvor jeg har givet mig selv 40.000 mønter og brugt 30.000 af dem på donuts i level 100. Der er på billedet endda illustreret dataene fra filen som gemmer spillet. 30.000 mønter er mere end rigeligt til at klare hele spillet, men i level 100 er baren knap rokket sig efter at have brugt så stor sum end penge.

(Billede 4)



*Graf 2* viser stigningen fra level 1 til 100 for en god ordens skyld. Her fås også et meget bedre indblik i hvor svært det kan blive at stige level og hvad det senere henne vil kræve.

  
(Graf 2)

# Konklusion

# Bilag

**Alt kode til mit program, inddelt efter tabs i Processing (1 tabel er 1 Processing tab).**

|  |
| --- |
| **[Main]**  import ddf.minim.\*;  Minim minim;  int index = 0;  AudioSample[] songs = new AudioSample[11];  // Need G4P library  import g4p\_controls.\*;  CutieClass cutie = new CutieClass();  PImage bubble;  int valuta = 0;  float cutieClick = 1;  float icecream = 10;  float lollipop = 25;  float donut = 50;  float meter = 0;  //174 ganget med cutie.lvl opløftet i 1.5  float xpTilLevel = 174;  public void **setup**() {    size(350, 350, JAVA2D);    createGUI();    customGUI();    gameLoad();    // Place your setup code here    cutie.loadImages();    minim = new Minim(this);    songs[0] = minim.loadSample("sound1.wav");    songs[1] = minim.loadSample("sound2.wav");    songs[2] = minim.loadSample("sound3.wav");    songs[3] = minim.loadSample("sound4.wav");    songs[4] = minim.loadSample("sound5.wav");    songs[5] = minim.loadSample("sound6.wav");    songs[6] = minim.loadSample("sound7.wav");    songs[7] = minim.loadSample("sound8.wav");    songs[8] = minim.loadSample("sound9.wav");    songs[9] = minim.loadSample("sound10.wav");    songs[10] = minim.loadSample("sound11.wav");    bubble = loadImage ("edited talebobbel.png");  }  public void **draw**() {    background(255);    //Potensfunktion    xpTilLevel = 174\*pow(cutie.lvl+1, 1.5);    stroke(255, 10, 10);    rect(150, 190, 176, 25);    noFill();    noStroke();    //Love-o-meter    fill(0, 102, 153);    textSize(18);    text("Love-o-meter", 175, 175);    noFill();    //Candy Shop    fill(0, 102, 153);    textSize(18);    text("Candy Shop", 182, 250);    noFill();    //CurrencyText    fill(0, 102, 153);    textSize(18);    text("Currency", 193, 115);    noFill();    //Valuta    fill(0, 12, 133);    textSize(12);    text(valuta, 198, 137);    noFill();    //Kalder på lvlUP funktionen    cutie.lvlUP();    //Filling love-o-meter    fill(255, 0, 170);    float barLength = floor((174/xpTilLevel)\*meter);    rect(151, 191, barLength, 24);    noFill();    noStroke();    //Talebobbel    image (bubble, 125, 0, 226, 101);    //Kalder på draw funktionen    cutie.**draw**();    //Winning Screen    if (cutie.lvl > 5) {      WinningScreen();    }  }  //Kalder på vores coins funktion og sørger for at vi kan klikke på karakteren for coins.  void **mousePressed**() {    cutie.coins();  }  void gameSave() {    PrintWriter saveWriter = createWriter("CutieSave.txt");    //skriver til bufferen    saveWriter.print(cutie.lvl +";"+ meter +";"+ valuta);    //flush gemmer bufferen til filen.    saveWriter.flush();    saveWriter.close();  }  void gameLoad() {    String data;    BufferedReader saveReader = createReader("CutieSave.txt");    try {      data = saveReader.readLine();    }    catch (IOException e) {      e.printStackTrace();      data = null;    }    if (data == null) {      cutie.lvl = 0;      meter = 0;      valuta = 0;    } else {      String[] variables = split(data, &apos;;&apos;);      cutie.lvl = int (variables[0]);      meter = float (variables[1]);      valuta = int (variables[2]);    }  }  // Use this method to add additional statements  // to customise the GUI controls  public void customGUI() {  } |

|  |
| --- |
| **[CutieClass]**  class CutieClass {    //Arraylist som holder billederne    PImage[] characters = new PImage[6];    int lvl = 0;    int reset = 0;    CutieClass()    {    }    //Loader billeder og angiver startsværdi 0. Vi fortæller den at den kan plusse op til næste level under visse omstændigheder    void loadImages()    {      for (int x = 0; x < 6; x++) {        int y = x + 1;        characters[x] = loadImage("lv" + y + ".png");      }    }    //Vi går her én level op ved at plusse level med et    void nextLvl()    {      lvl = lvl + 1;    }    //Vi angiver position og tegner vores karakter her    void **draw**()    {      if (lvl > 5) {        image(characters[5], 0, 0);      } else {        image(characters[lvl], 0, 0);      }    }    //Coin funktion som tjekker om vi er inden for vores billedes ramme, hvis vi er det får vi coins.    void coins() {      PImage img;      if (lvl > 5) {        img = characters[5];      } else {        img = characters[lvl];      }      if ((mouseX > img.X && mouseX < img.X+img.width) && (mouseY > img.Y && mouseY < img.Y + img.height)) {        valuta += cutieClick;        index = int(random(songs.length));        songs[index].trigger();        cursor(HAND);      }    }    void lvlUP() {      if (meter >= xpTilLevel) {        cutie.nextLvl();        meter = reset;      }    }  } |

|  |
| --- |
| **[WinningScreen]**  void WinningScreen()  {    textSize(16);    text("You&apos;ve won,", 200, 35);    text("but you can still play!", 163, 55);  } |

|  |
| --- |
| **[gui]**  /\* =========================================================   \* ====                   WARNING                        ===   \* =========================================================   \* The code in this tab has been generated from the GUI form   \* designer and care should be taken when editing this file.   \* Only add/edit code inside the event handlers i.e. only   \* use lines between the matching comment tags. e.g.   void myBtnEvents(GButton button) { //\_CODE\_:button1:12356:       // It is safe to enter your event code here   } //\_CODE\_:button1:12356:     \* Do not rename this tab!   \* =========================================================   \*/  public void imgButton2\_click1(GImageButton source, GEvent event) { //\_CODE\_:imgButton2:501732:    println("imgButton2 - GImageButton >> GEvent." + event + " @ " + millis());    if (valuta >= 50) {      valuta -= donut;      meter += donut;    } else {      println("You do not have enough coins for a donut!");    }  } //\_CODE\_:imgButton2:501732:  public void imgButton3\_click1(GImageButton source, GEvent event) { //\_CODE\_:imgButton3:400537:    println("imgButton3 - GImageButton >> GEvent." + event + " @ " + millis());    if (valuta >= 25) {      valuta -= lollipop;      meter += lollipop;    } else {      println("You do not have enough coins for a lollipop!");    }  } //\_CODE\_:imgButton3:400537:  public void imgButton4\_click1(GImageButton source, GEvent event) { //\_CODE\_:imgButton4:987310:    println("imgButton4 - GImageButton >> GEvent." + event + " @ " + millis());    if (valuta >= 10) {      valuta -= icecream;      meter += icecream;    } else {      println("You do not have enough coins for an icecream!");      //text("You do not have enough coins for an icecream!", 125, 50);    }  } //\_CODE\_:imgButton4:987310:  public void imgButton5\_click1(GImageButton source, GEvent event) { //\_CODE\_:imgButton5:237760:    println("imgButton5 - GImageButton >> GEvent." + event + " @ " + millis());  } //\_CODE\_:imgButton5:237760:  public void imgButton7\_click1(GImageButton source, GEvent event) { //\_CODE\_:imgButton7:885239:    println("imgButton7 - GImageButton >> GEvent." + event + " @ " + millis());  } //\_CODE\_:imgButton7:885239:  public void imgButton8\_click1(GImageButton source, GEvent event) { //\_CODE\_:imgButton8:686714:    println("imgButton8 - GImageButton >> GEvent." + event + " @ " + millis());  } //\_CODE\_:imgButton8:686714:  public void imgButton9\_click1(GImageButton source, GEvent event) { //\_CODE\_:imgButton9:632010:    println("imgButton9 - GImageButton >> GEvent." + event + " @ " + millis());  } //\_CODE\_:imgButton9:632010:  public void button1\_click1(GButton source, GEvent event) { //\_CODE\_:button1:258020:    println("button1 - GButton >> GEvent." + event + " @ " + millis());    gameSave();  } //\_CODE\_:button1:258020:  // Create all the GUI controls.  // autogenerated do not edit  public void createGUI(){    G4P.messagesEnabled(false);    G4P.setGlobalColorScheme(GCScheme.BLUE\_SCHEME);    G4P.setCursor(ARROW);    surface.setTitle("Raise a cutie");    imgButton2 = new GImageButton(this, 267, 268, 50, 50, new String[] { "edited donut.png", "edited donut.png", "edited donut.png" } );    imgButton2.addEventHandler(this, "imgButton2\_click1");    imgButton3 = new GImageButton(this, 208, 268, 50, 50, new String[] { "edited lollipop.png", "edited lollipop.png", "edited lollipop.png" } );    imgButton3.addEventHandler(this, "imgButton3\_click1");    imgButton4 = new GImageButton(this, 147, 267, 50, 50, new String[] { "edited ice cream.png", "edited ice cream.png", "edited ice cream.png" } );    imgButton4.addEventHandler(this, "imgButton4\_click1");    imgButton5 = new GImageButton(this, 238, 125, 15, 15, new String[] { "edited coin.png", "edited coin.png", "edited coin.png" } );    imgButton5.addEventHandler(this, "imgButton5\_click1");    label2 = new GLabel(this, 153, 322, 23, 20);    label2.setText("10");    label2.setTextBold();    label2.setOpaque(false);    label3 = new GLabel(this, 214, 323, 28, 20);    label3.setText("25");    label3.setTextBold();    label3.setOpaque(false);    label4 = new GLabel(this, 274, 322, 22, 20);    label4.setText("50");    label4.setTextBold();    label4.setOpaque(false);    imgButton7 = new GImageButton(this, 173, 325, 15, 15, new String[] { "edited coin.png", "edited coin.png", "edited coin.png" } );    imgButton7.addEventHandler(this, "imgButton7\_click1");    imgButton8 = new GImageButton(this, 238, 326, 15, 15, new String[] { "edited coin.png", "edited coin.png", "edited coin.png" } );    imgButton8.addEventHandler(this, "imgButton8\_click1");    imgButton9 = new GImageButton(this, 295, 325, 15, 15, new String[] { "edited coin.png", "edited coin.png", "edited coin.png" } );    imgButton9.addEventHandler(this, "imgButton9\_click1");    button1 = new GButton(this, 28, 316, 80, 30);    button1.setText("Save Game");    button1.addEventHandler(this, "button1\_click1");  }  // Variable declarations  // autogenerated do not edit  GImageButton imgButton2;  GImageButton imgButton3;  GImageButton imgButton4;  GImageButton imgButton5;  GLabel label2;  GLabel label3;  GLabel label4;  GImageButton imgButton7;  GImageButton imgButton8;  GImageButton imgButton9;  GButton button1; |