2019

ÅRSPRØVE
PROGRAMMERING - B

Andreas Petersen

Odense tekniske Gymnasium

14. maj 2019

COPS & ROBBERS

Denne rapport indeholder en beskrivelse af en simulation af ”politi og røvere” baseret på nogle forudbestemte krav og en opfølgning på programmets nuværende stadie.

Titelblad

Vejleder
Søren Præstegaard

Fag
Programmering B

Klasse og skole
2.D - Odense Tekniske Gymnasium

Antal tegn
15390 (6.4 normalsider)

Dato
14. maj 2019

Indholdsfortegnelse

Indledning

Politi og røvere er en gammel leg, som mange børn har leget gennem tiden. Det er en leg, hvor en røver forsøger at slippe væk fra politiet uden at blive fanget. Denne leg har et sæt specifikke regler, som for børn er lette at følge. Disse regler kan også skrives ind i et program, som så kan simulere legen i en computer. I simuleringen kan der så laves justeringer, som gør røveren eller politiet bedre til henholdsvis at slippe fra politiet og til at fange røveren. Disse forbedringer kan man så tage ud og bruge i legen, men man kan også bare hygge sig med sin simulation og måske endda lave et spil ud af det.

Problemformulering

Når man skal lave et spil eller en simulation af legen ”Politi og røvere”, skal man have alle informationer på både røveren og politiet på plads. Man skal have deres placering, retning og hastighed. Følgende vil blive behandlet:

- En kort redegørelse for, hvordan røveren og politiet agerer i det udleverede program.
- Beskrivelse af optimerende elementer tilføjet til programmet.
- Analyse af et simulationsprogram og dets opførsel efter tilføjelsen af elementer.
    - Herunder en inddragelse af undersøgelser af metoder til at forbedre røverens- og politiets strategi.
- Diskussion af resultater fra analysen.
    - Overholder programmet kravene stillet.
- Perspektivering til fremtidige design og funktionalitet af programmet.

Kravspecifikation

I dette projekt har jeg valgt at fokusere på oplevelsen af programmet, og derfor har jeg gået i dybden med udvikling af billede og lyd af programmet. Heraf har jeg opstillet nogle krav, som skal sikre, at programmet får et godt design og lydbillede:

Hårde krav

- Politiet og røveren skal have en figur hver, som repræsenterer dem.
- Der skal være en baggrundsmusik, som ligger under temaet ”politi og røvere”.

Bløde krav

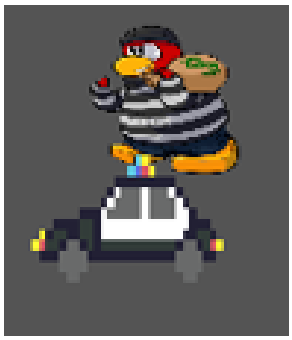
- Der skal være lydeffekter, som aktiveres, når henholdsvis røveren undslipper politiet eller politiet fanger røveren.
- Det skal være underholdende at se på
    - Der må godt gå et godt stykke tid, inden en af parterne vinder, så der sker noget undervejs.

Politiets- og røverens indledende strategi

I programmet vi fik udleveret, flygtede røveren fra politiets massemidtpunkt, og politiet direkte efter røverens aktuelle position. Denne strategi var ikke optimal for begge parter, hvorfor det var vores opgave at finde på nogle forbedringer til det. Derudover, som nævnt under kravspecifikation, var målet for mit projekt også at lave nogle oplevelsesmæssige optimeringer til spillet. Disse forbedrende elementer er beskrevet herunder.

Optimerende elementer

Udover at forbedre røverens- og politiets strategi (dette blive gennemgået i afsnittet om algoritmer), så ville jeg gerne have, at røveren og politiet hver i sær havde deres egnet karakter, så man med det samme kunne se, hvem der var hvem. Derfor var jeg inde og finde et billede af en tegnet røver og for politiet en pixeleret politibil (se figur 1).



Figur 1

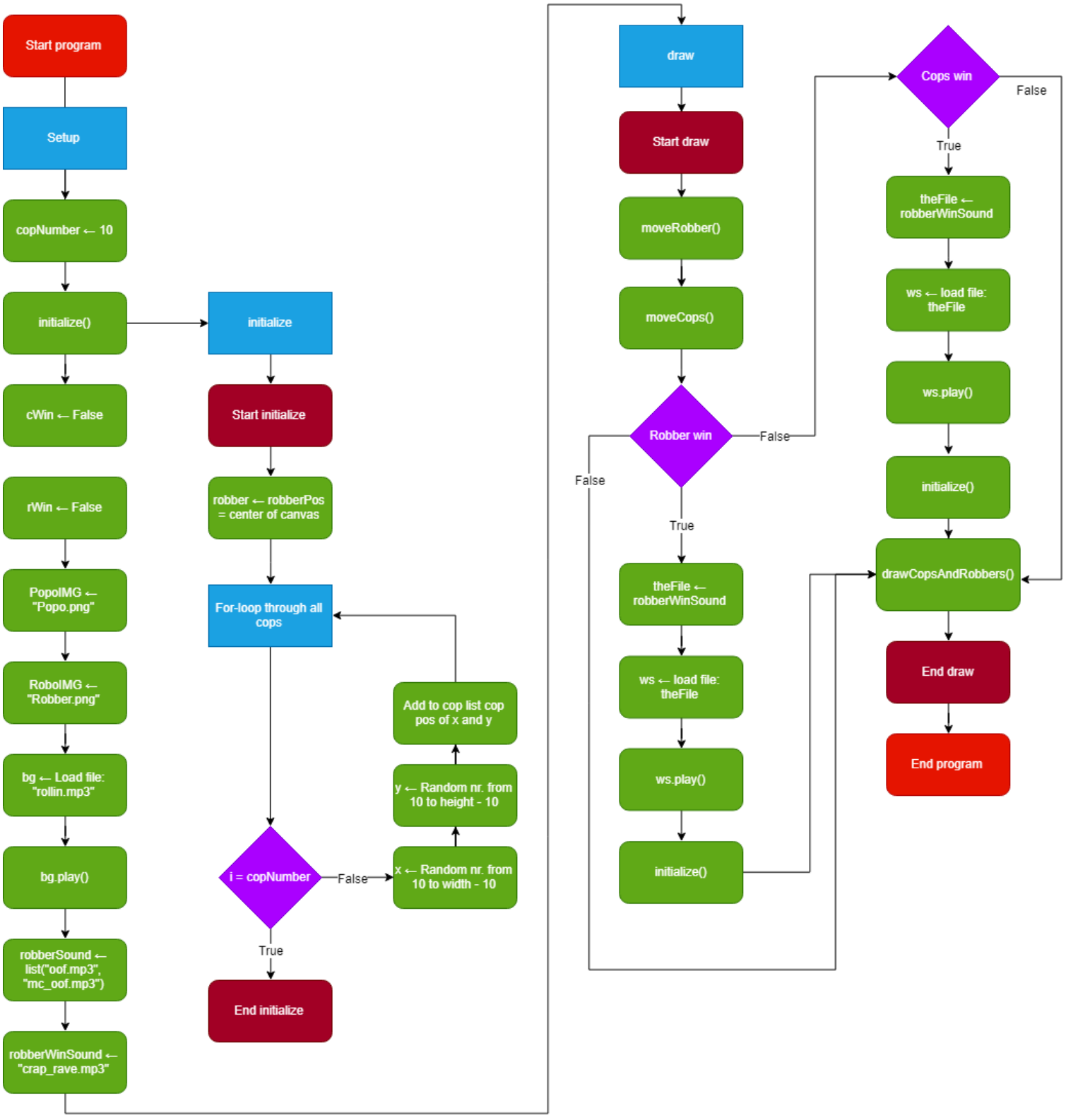
Lydbilledet skulle der også optimeres på, da programmet i sit udleverede stadie var dødstille. Derfor besluttede jeg mig for, som nævnt under krav, at der skulle være noget baggrundsmusik og også nogle forskellige lydeffekter for at gøre det mere interessant.

Programmets opbygning

I dette afsnit bliver programmets opbygning gennemgået.

Flowdiagram

Herunder er der lavet et flowdiagram, som overordnet set beskriver programmets funktion.



Figur 2 Flowdiagram over programmets overordnede funktion.

Beskrivelse af algoritmer

I dette afsnit bliver programmets algoritmer og deres effekt beskrevet.

Røverens bevægelse

For at røveren skal kunne slippe væk fra politiet, skal den have en måde at vide, hvor politiet er, så den ikke bare løber ligefrem ind i politiet. Dette kan løses ved brug af en algoritme. Den ummildbare mest effektive måde at slippe fra politiet er, at finde politiets massemidtpunkt, og derefter bestemme røverens retning væk fra politiet. Udover at have politiets massemidtpunkt, skal røveren også gange det punkt med en vægtning således, at røveren hverken bliver presset fra alle retninger resulterende i, at røveren står stille, hverken skal den vægtes for lavt, så den ikke afviger fra politiet.

Denne vægtning skulle findes, og dette var ikke en nem sag. Det tog tid og forsøg med flere forskellige metoder.

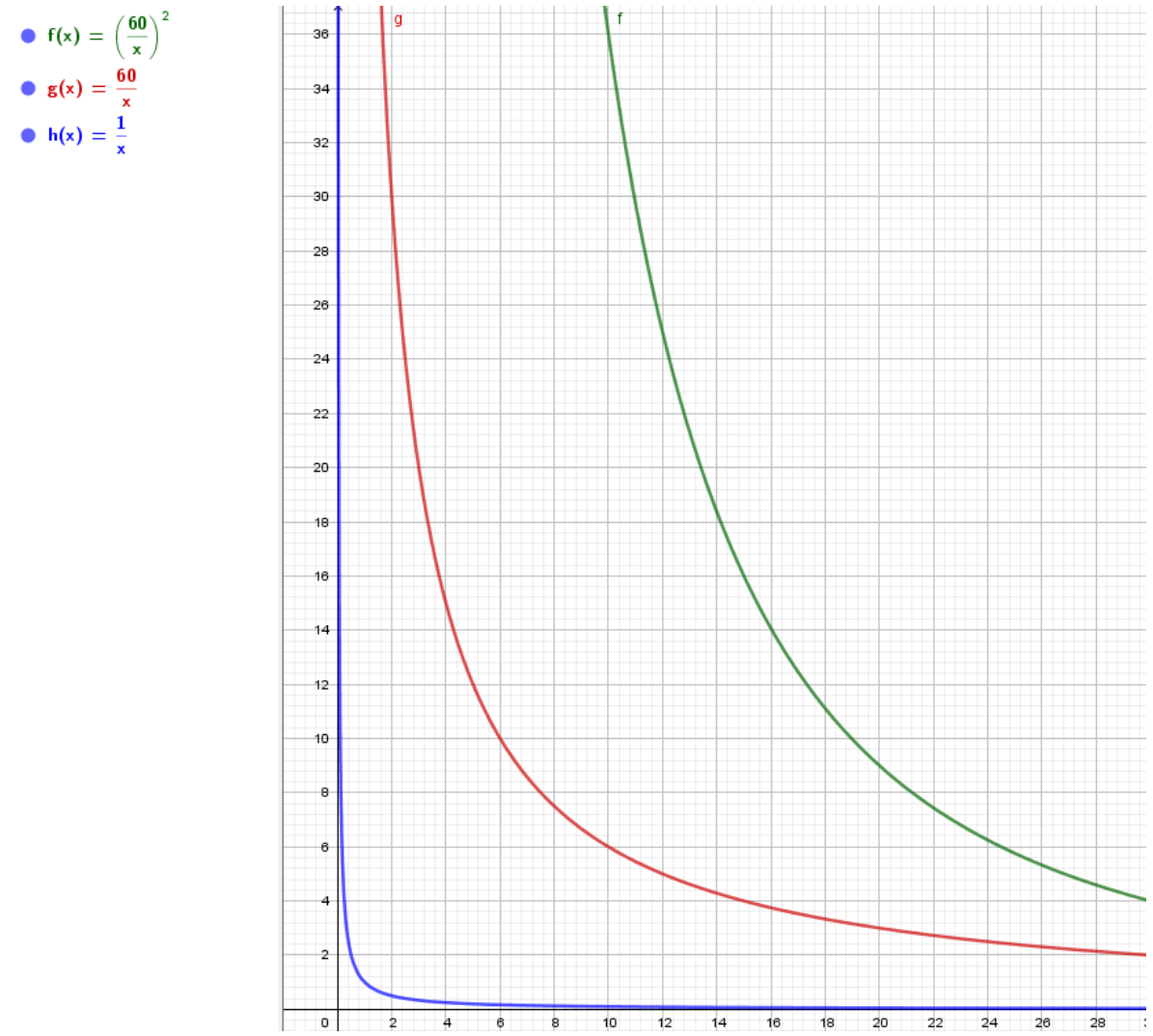
Det var vigtigt, at vægtningen skubbede mere væk fra det nærmeste politi end det fjerneste. Derfor var det første forsøg med en hyperbel $\frac{1}{distance}$. Det virkede principielt, men det var ikke nok til at afvige væk fra politiet. I det andet forsøg blev der brugt $\frac{60}{distance}$, hvilket virkede bedre, men var stadig ikke helt nok til at være tilfredsstillende. Det sidste forsøg var med $\left(\frac{60}{distance}\right)^2$.

På figur 3, linje 4 er vægtningen sat ind i programmet. Linjerne i kodeeksemplerne gemmen rapporten passer ikke med linjerne i selve programmet, men de er der for at hjælpe med at vise, hvad der bliver henvist til.

```
1 totalWeight = 0
2     for cop in cops:
3         distance = dist(cop.pos.x, cop.pos.y, robber.pos.x, robber.pos.y)
4         w = (60/distance)**2
5 
6         totalWeight += w
7         COM.x += cop.pos.x * w
8         COM.y += cop.pos.y * w
```

Figur 3 Vægtningen af politiets massemidtpunkt for senere brug af flytning af røveren.

$\left(\frac{60}{distance}\right)^2$ var den bedste vægtning af dem alle, da det ikke var for meget eller for lidt. Indimellem disse værdier har der været andre, som langsomt har nærmet sig den godkendte vægtning. På figur 4 ses en illustration af de forskellige metoder af vægtning.



Figur 4 Illustration af forskellige vægtningsmetoder.

For hvert billede (frame) der går, vil denne algoritme blive brugt, og gemt for at bruges til at flytte røveren i retningen væk fra politiet.

Politiets bevægelse

Politiets oprindelige funktion var med en hastighed på 1 (den samme hastighed som røveren), hele tiden at køre mod røverens aktuelle position. Dette, efter at have optimeret røveren, var ikke en ret god strategi, da røveren i gennemsnit vandt 99% af gangene. Der skulle altså noget andet til. Det ville være oplagt, hvis politiet kunne forsøge at komme foran røveren for at være på forkant med røverens bevægelser, men hvordan?...

Det som endte med at blive brugt, var at røverens retning (retningsvektor) blev gemt som en egenskab af røveren kaldet ”last” (robber.last). Den retningsvektor blev så for hver politimand skaleret med afstanden fra politiet til røveren i det frame (igen bliver dette kørt

igennem for hvert frame). Politiet ville altså for hver frame forsøge at bevæge med den mellemliggende afstand for røveren og politiet foran den retning, som røveren havde. Det resulterede i, at politiet kørte hen foran røveren og ventede på, at den skulle komme og blive fanget.

Beskrivelse af vigtige funktioner

Der er flere funktioner som er centrale i at sikre programmets funktion. I dette afsnit er disse funktioner medtaget. I og med at funktionerne: setup, initialize og draw kort er blevet beskrevet i flowdiagrammet, og at funktionerne: moveCops og moveRobber er beskrevet under ”Beskrivelse af algoritmer”, vil de ikke indgå i dette afsnit.

Info()

Denne funktion (se figur 5) tager som input enten et 1 eller 2. Hvis inputtet (”winner”) er 1, så er det, fordi røveren har vundet (linje 2), og derfor skal teksten illustreres for røveren og omvendt for politiet (linje 11). Begge vindertilfælde er opskrevet på samme måde; hver har en farve defineret med fill-funktionen (f.eks. linje 3), og hver har nogle linjer med tekst indsat med text-funktionen (f.eks. linje 4). fill-funktionen tager tre tal som input, dette står for RGB (red green blue). text-funktionen tager tre inputs. Det første input er en tekststreng, dog kan det også være tal. De sidste to inputs er x- og y-koordinater for placeringen af texten.

```
1  def info(winner):
2      if winner == 1:
3          fill(0, 255, 0)
4          text('Robber points {}'.format(rCounter), 10, 15)
5          fill(255, 0, 0)
6          text('Cops points {}'.format(cCounter), width-120, 15)
7          fill(255, 255, 255)
8          percent = (float(rCounter)/(float(rCounter)+float(cCounter)))*100
9          percent = "{:10.2f}".format(percent)
10         text('{}%'.format(percent), width/2 - 40, 15)
11     if winner == 2:
12         fill(0, 255, 0)
13         text('Cops points {}'.format(cCounter), width-121, 15)
14         fill(255, 0, 0)
15         text('Robber points {}'.format(rCounter), 10, 15)
16         fill(255, 255, 255)
17         percent = (float(rCounter)/(float(rCounter)+float(cCounter)))*100
18         percent = "{:10.2f}".format(percent)
19         text('{}%'.format(percent), width/2 - 40, 15)
```

Figur 5 Funktionen for fremvisningen af informationer som point og procent.

robberWin()

Denne funktion (se figur 6) tager røveren som input, så den ved, hvad røveren er for en ting og kender til de egenskaber, som den har (egenskaberne bliver gennemgået i afsnittet ”Beskrivelse af centrale klasser”). Den sikrer først ved at definere ”checker” til False (falsk. Linje 4), at hvis røveren ikke har kriterierne for at vinde, vil den returnere False (linje 7), og dermed vise programmet, der hvor funktionen er brugt i programmet, at røveren har tabt. For at vinde, skal røveren (med røveren menes der dimensionerne af det billede som udgør røveren, som kan ses, når programmet køres) ramme en af vinduets fire kanter (linje 5). Hvis røveren så gør det, vil ”checker” være lig med True (sand, linje 6) og programmet ved dermed, at røveren har vundet.

```
1 def robberWin(robber):
2     # Denne funktion skal returnere True,
3     # hvis røveren er nået udenfor skærmen
4     checker = False
5     if robber.pos.y + roboH >= height or robber.pos.x <= 0 or robber.pos.y <= 0 or robber.pos.x + roboW >= width:
6         checker = True
7     return checker
```

Figur 6 Funktionen for bestemmelsen af, hvornår røveren vinder spillet.

copsWin()

Denne funktion (se figur 7) minder meget om den beskrevet over. Der er dog nogle forskelle, hvilket beskrives i dette afsnit. For det første er det et andet input til denne funktion end røveren også, nemlig politiet. Dette er af samme årsager som for røveren, at politiets egenskaber skal kendes af funktionen. I og med at politiet ikke kun er en person men en liste af politi, så skal der ses igennem alle politi, for at undersøge om bare en af dem har vinderkriteriet (for-loop, linje 5 til 9). Når hver politibetjent bliver tjekket for, tjekkes der efter, hvorvidt de er tæt nok på røveren til at blive kategoriseret som at have fanget røveren. Dette gøres ved først at definere afstanden fra politiet til røveren (linje 6) og derefter bruge den afstand til at tjekke om den afstand er mindre end halvdelen af billede størrelserne (billede størrelserne værende størrelserne på de billeder, som udgør politiets og røverens synlige fremstillinger i programmet. Der tjekkes på linje 5).

```
1 def copsWin(cops, robber):
2     # Denne funktion skal returnere True,
3     # hvis politiet har fanget røveren
4     checker = False
5     for cop in cops:
6         distance = dist(cop.pos.x, cop.pos.y, robber.pos.x, robber.pos.y)
7         if distance < IMGsize/2:
8             checker = True
9     return checker
```

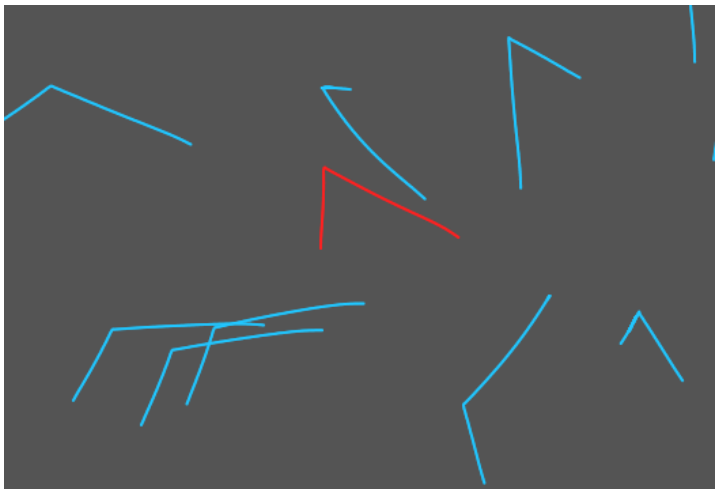
Figur 7 Funktionen for bestemmelsen af, hvornår politiet vinder spillet.

drawCopsAndRobbers()

Denne funktion (se figur 8) forklarer programmet, hvordan den skal vise røveren og politiet. For hver gang funktionen bliver kørt, sættes baggrunden til at være mørke grå (linje 2). Dette gøres, for at fjerne alle tidligere tegninger fra vinduet. Funktionen tager også et tidligere defineret vinder nummer (1 for røver og 2 for politi, som beskrevet tidligere) og bruger det i info-funktionen. De udkommenterede linjer (linje 5-6 & 10-12) er, hvordan programmet startede med at være (se figur 9 for illustration af første version af programmet). Her var det en ellipse med røverens farve taget fra røverens egenskaber (robber.col). Ellipsen havde røverens position og fyldte en pixel. Det samme gjaldt for politiet. I stedet for bruger programmet nu funktionen image til at fremvise billederne for røveren og for politiet (linje 7 & 13).

```
1  def drawCopsAndRobbers():
2      background(60, 67, 79)
3      info(winNumber)
4
5      # fill(robber.col)
6      # ellipse(robber.pos.x, robber.pos.y, 1, 1)
7      image(RoboIMG, robber.pos.x, robber.pos.y, roboW, roboH)
8
9      for cop in cops:
10         # noStroke()
11         # fill(cop.col)
12         # ellipse(cop.pos.x, cop.pos.y, 1, 1)
13         image(PopoIMG, cop.pos.x, cop.pos.y, popoW, popoH)
```

Figur 8 Funktionen for, hvordan programmet skal tegne røveren og politiet i vinduet.



Figur 9 Illustration af programmet inden billeder af røver og politi blev tilføjet.

Beskrivelse af centrale klasser

Til programmet er der to klasser i brug. Klasser er en form for skabeloner, som skaber objekter. Objekter er ting, som har nogle attributter (egenskaber). Hvis man f.eks. har klassen MOR, så kan den producere nogle objekter, mennesker. Hvert menneske har så nogle beskrivende egenskaber som f.eks. kunne have en mor, en far, to hænder, et job og meget mere. På samme måde bruges klasser i dette program, dog ikke med mennesker. Her bruges klasserne ”Cop” og ”Robber” til at skabe tilfælde af røveren og af politiet.

Politiets klasse

Politiet har 3 egenskaber (se figur 10). Egenskaberne er defineret under en metode kaldet __init__ (linje 1-5). __init__ tager 3 parametre. Det første parameter er ”self”. Self repræsenterer det tilfælde af objektet selv. De andre to parametre er x og y, som skrives ind, når de skal bruges til i dette tilfælde at sætte positionen af politiet (gøres ved ”Cop(x, y)”). Men hvad laver den ”self” så?... Når klassen bliver nævnt (Cop(1, 2)), opretter Python et objekt og sætter det ind som det første parameter ”self” i metoden ”__init__”. Herefter får objektet så egenskaberne: pos, speed og col.

Den første egenskab for politiet er positionen af politiet angivet som en vektor (linje 3). Den anden egenskab er hastigheden, som er sat til at være 1 (linje 4). Den sidste egenskab for politiet er farven, som er sat til at være blå (linje 5).

```
1 class Cop:
2     def __init__(self, x, y):
3         self.pos = PVector(x, y)
4         self.speed = 1
5         self.col = color(33, 198, 255)
```

Figur 10

Røverens klasse

Røverens klasse (se figur 11) har mange af de samme egenskaber, som ikke gennemgås her, men der er en forskel. Røveren har en egenskab kaldet ”last” (linje 6). Last bruges til at logge den sidste retning af røveren (brugen af retningsvektoren for røveren er beskrevet under ” Beskrivelse af algoritmer”).

```
1 class Robber:
2     def __init__(self, x, y):
3         self.pos = PVector(x, y)
4         self.speed = 1
5         self.col = color(255, 33, 33)
6         self.last = PVector(0, 0)
```

Figur 11

Beskrivelse af anvendte biblioteker

Programmet bruger også flere forskellige biblioteker. I dette afsnit er bibliotekerne i brug beskrevet.

Minim [1]

Minim er ikke et standardbibliotek tilgængelig i Python. Det er et bibliotek, som er særligt for Processing, og som skal importeres på en særlig måde (se bilag 1 for importering af ”minim”). Minim er et bibliotek, som kan hjælpe med at afspille filer i Processing. Dette kan gøres ved at bruge funktioner som loadFile og play.

Time

Time er et bibliotek, som har en masse funktioner under emnet tid. Den kan bl.a. give programmet informationer omkring klokken. Dog er det ikke det, som biblioteket bruges til i dette program. Her bruges det med funktionen sleep til at sætte programmet på pause. Denne funktion har ikke effekt på funktionerne i minim, hvilket gør, at musikken kan fortsætte med at spille, mens programmet er på pause. Med denne effekt sættes spillet på pause, når røverens vindermusik spiller, så man kan høre den, uden at der bliver aktiveret andre lyde i baggrunden.

[1] Dokumentationen for minim er fra denne hjemmeside (Minim, Opdateret 05. Juli 2018)

Overholdelse af krav

Hårde krav

Politiet og røveren skal have en figur hver, som repræsenterer dem

Både politiet og røveren har fået figurer, som repræsenterer dem. Røveren har fået en tegneseriefigur af en røver og politiet har fået en politibil hver.

Der skal være en baggrundsmusik, som ligger under temaet ”politi og røvere”

I baggrunden spiller ” Chamillionaire - Ridin' ft. Krayzie Bone”, hvilket omhandler en gruppe kriminelle, som er imod alt hvad der hedder politi. Sangen ligger derfor i den grad under temaet ”politi og røvere”.

Bløde krav

Der skal være lydeffekter, som aktiveres, når henholdsvis røveren undslipper politiet eller politiet fanger røveren.

For både røveren og politiet er der tilføjet lydeffekter, som understrejer at den pågældende part har vundet. Når politiet vinder afgiver røveren nogle stønne lyde, som tegn på tab. Når røveren vinder afspilles et fedt beat, da røveren skal fejre det succesfulde røveri.

Det skal være underholdende at se på

Der var ikke så lang tid tilbage, så her nåede jeg ikke så langt, som jeg villa have. Det var meningen, at både karaktererne og baggrunden skulle være animeret, men dette kan så være en fremtidig forbedring til programmet i stedet for.

Forslag til forbedringer

Som sagt under ”overholdelse af krav”, er en forbedring til programmet et re-design af det visuelle. Her kunne det være fedt, hvis røveren og politiet bar animeret med røverens ben og bilernes hjul. Baggrunden kunne være et vejsystem, som ændrede sig i forhold til røveren.

En anden forbedring ville kunne være en mere visuel repræsentation af de to vindende stadier af programmet, så man mere tydeligt ville kunne se, hvem der havde vundet.

Man kunne også lave nogle forbedringer i form af nogle justeringer af det nuværende program. F.eks. så afspiller baggrundsmusikken samtidig med, at røverens vindermusik afspilles. Dette og andet kunne justeres i fremtiden.

Skulle man tilføje nogle funktioner til programmet, så kunne man give røveren nogle superkræfter. Dette kunne være at fryse politiet i 2 sekunder. Dette ville gøre det mere fair for røveren, siden politiet på nuværende tidspunkt er for god til at fange røveren.

Som et sidste eksempel på en forbedring til programmet, kunne man tilføje en startmenu, hvor man ville kunne lave ændringer som f.eks. antallet af politi inden spillet startes.

Arbejdsprocessen

Arbejdsprocessen har været meget ligefrem. Vi havde til at starte med nogle opgaver som at ændre i funktionen moveRobber, eller at lave en bedre strategi for politiet. Efter at have lavet opgaverne, skulle vi så selv finde på ting, som vi ønskede at vores program skulle. Disse elementer, som jeg har implementeret (et af dem værende afspildning af musik), er noget, som jeg har tænkt: ”Det kunne være fedt at have i programmet”, og derefter har implementeret det i programmet.

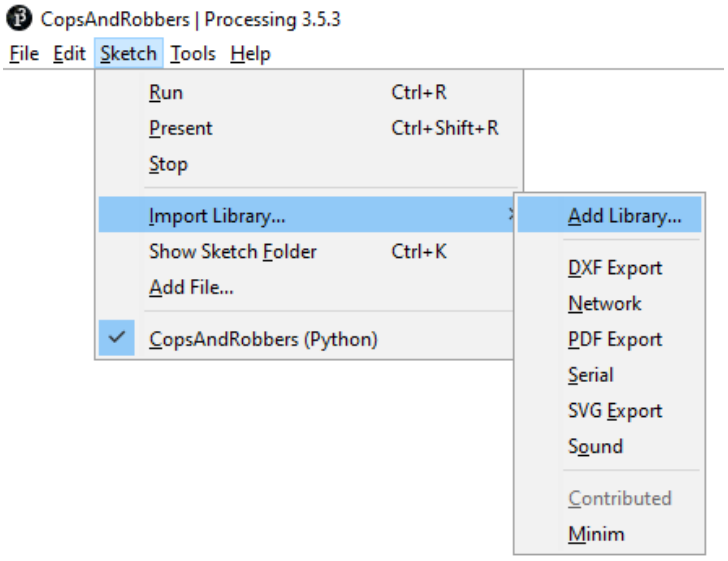
Referencer

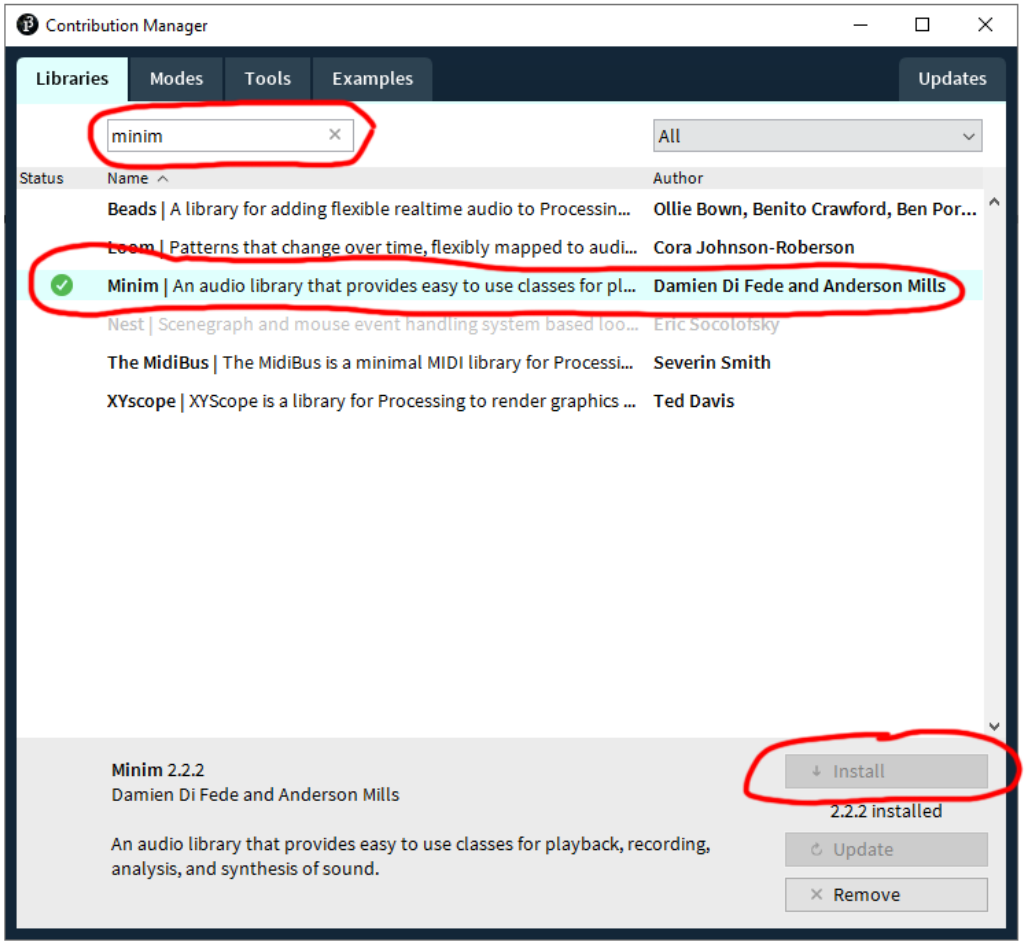
Minim. (Opdateret 05. Juli 2018). Hentet fra code.compartmental:
http://code.compartmental.net/minim/

Bilag

Bilag 1 - indportering af minim

Minim kan installeres vha. billederne herunder. Efter at have installeret minim, kan biblioteket bruges ved at skrive ”add_library('minim')” i toppen af programmet.





Bilag 2 - kildekode

CopsAndRobbers.pyde

```
from random import randint
import time
from RobberLib import *
from random import randint
import time
from RobberLib import *
from CopLib import *
add_library('minim')


def setup():
    size(1000, 1000)
    global cops, cWin, rWin, cCounter, rCounter, counter, copNumber,
afstandList, RoboIMG, PopoIMG, popoH, popoW, roboW, roboH, IMGsize,
winNumber, minim, robberSound, robberWinSound
    fps = 1000
    copNumber = 10
    cops = list()
    initialize()
    textSize(15)
    cWin = False
    rWin = False
    cCounter = 0
    rCounter = 0
    counter = 0
    afstandList = []
    frameRate(fps)
    PopoIMG = loadImage("Popo.png")
    RoboIMG = loadImage("Robber.png")
    IMGsize = 60
    popoH = IMGsize * 0.8
    popoW = IMGsize * 1.2
    roboW = IMGsize
    roboH = IMGsize
    winNumber = 0
    minim = Minim(this)
    bg = minim.loadFile("rollin.mp3")
    bg.play()
    bg.shiftGain(bg.getGain(), -15, 2500)
    robberSound = ["oof.mp3", "mc_oof.mp3"]
    robberWinSound = ["crap_rave.mp3"]


def initialize(rPos=False):
    # I denne funktion skal hele spillet
    # startes forfra.
```

```
    global cops, robber, running, copNumber
    # Bagrunden cleares
    background(84, 84, 84)

    running = True

    # Ny position til røveren
    if rPos:
        robber = Robber(rPos.x, rPos.y)
    else:
        robber = Robber(width/2, height/2)

    # Nye positioner til politiet.
    # Tøm listen
    cops[:] = []
    # og tilføj nye politimænd
    for i in range(0, copNumber):
        x = random(10, width - 10)
        y = random(10, height - 10)
        cops.append(Cop(x, y))


def draw():
    global cWin, rWin, winNumber
    rWin = False
    cWin = False

    if running:
        moveRobber()
        moveCops()

    if robberWin(robber):
        # Her kan du vælge hvad der skal ske
        # hvis røveren vinder
        if running:
            print("The robber wins")
        # running = False
        rWin = True
        percent = winPercentage(cWin, rWin)
        if percent[0]:
            print(percent[1])
        theFile = robberWinSound[0]
        ws = minim.loadFile(theFile)
        ws.play()
        langde = float(ws.length()/1000)
        time.sleep(langde)
        initialize()
        winNumber = 1
        info(winNumber)
```

```
    if copsWin(cops, robber):
        # Her kan du vælge hvad der skal ske
        # hvis politiet vinder
        if running:
            print("The Cops wins")
        # running = False
        cWin = True
        percent = winPercentage(cWin, rWin)
        if percent[0]:
            print(percent[1])
        theFile = robberSound[randint(0, 1)]
        ws = minim.loadFile(theFile)
        ws.play()
        langde = float(ws.length()/1000)
        time.sleep(langde)
        initialize()
        winNumber = 2
        info(winNumber)

    total = float(rCounter)+float(cCounter)
    fill(255, 255, 255)
    text('Total: {}'.format(int(total)), 4, height-4)

    drawCopsAndRobbers()


def info(winner):
    if winner == 1:
        fill(0, 255, 0)
        text('Robber points {}'.format(rCounter), 10, 15)
        fill(255, 0, 0)
        text('Cops points {}'.format(cCounter), width-120, 15)
        fill(255, 255, 255)
        percent =
(float(rCounter)/(float(rCounter)+float(cCounter)))*100
        percent = "{:10.2f}".format(percent)
        text('{}%'.format(percent), width/2 - 40, 15)
    if winner == 2:
        fill(0, 255, 0)
        text('Cops points {}'.format(cCounter), width-121, 15)
        fill(255, 0, 0)
        text('Robber points {}'.format(rCounter), 10, 15)
        fill(255, 255, 255)
        percent =
(float(rCounter)/(float(rCounter)+float(cCounter)))*100
        percent = "{:10.2f}".format(percent)
        text('{}%'.format(percent), width/2 - 40, 15)
```

```
def winPercentage(cWin, rWin):
    global cCounter, rCounter, counter
    ready = False
    counter += 1
    cPercent = 0
    rPercent = 0
    string = ""
    if cWin:
        cCounter += 1
    if rWin:
        rCounter += 1
    total = cCounter + rCounter
    if total != 0:
        ready = True
        cPercent = (float(cCounter) / float(total)) * 100
        rPercent = (float(rCounter) / float(total)) * 100
    if ready:
        string = "The game has run: {} times \nPolice win: {}% \nRobber
wins {}%".format(
            counter, cPercent, rPercent)
    return ready, string


def robberWin(robber):
    # Denne funktion skal returnere True,
    # hvis røveren er nået udenfor skærmen
    checker = False
    if robber.pos.y + roboH >= height or robber.pos.x <= 0 or
robber.pos.y <= 0 or robber.pos.x + roboW >= width:
        checker = True
    return checker


def copsWin(cops, robber):
    # Denne funktion skal returnere True,
    # hvis politiet har fanget røveren
    checker = False
    for cop in cops:
        distance = dist(cop.pos.x, cop.pos.y, robber.pos.x,
robber.pos.y)
        if distance < IMGsize/2:
            checker = True
    return checker


def moveCops():
    # Politiet bevæger sig imod det sted hvor røveren er.
    # Det er nemlig sådan politet gør.
    for cop in cops:
```

```
        afstand = robber.pos.copy()
        afstand.sub(cop.pos)

        afstand += robber.last * \
            (dist(cop.pos.x, cop.pos.y, robber.pos.x, robber.pos.y))
        # Sørg for at politiet kun bevæger sig 1 pixel
        afstand.normalize()
        afstand.mult(cop.speed)

        cop.pos.add(afstand)


def moveRobber():
    global COM, robber, distList
    # Udregn massemidtpunkt for politiet.
    # COM = Center of Mass
    COM = PVector()

    totalWeight = 0
    for cop in cops:
        distance = dist(cop.pos.x, cop.pos.y, robber.pos.x,
robber.pos.y)
        w = (60/distance)**2

        totalWeight += w
        COM.x += cop.pos.x * w
        COM.y += cop.pos.y * w

    COM.x /= totalWeight
    COM.y /= totalWeight

    # Find retningen væk fra COM.
    # Beregn afstanden til massemidtpunktet
    afstand = robber.pos.copy()
    afstand.sub(COM)

    # Vælg hvor langt røveren skal bevæge sig
    afstand.normalize()
    afstand.mult(robber.speed)

    # Flyt røveren
    robber.pos.add(afstand)
    robber.last = afstand


def drawCopsAndRobbers():
    background(84, 84, 84)
    info(winNumber)
```

```
    # fill(robber.col)
    # ellipse(robber.pos.x, robber.pos.y, 1, 1)
    image(RoboIMG, robber.pos.x, robber.pos.y, roboW, roboH)

    for cop in cops:
        # noStroke()
        # fill(cop.col)
        # ellipse(cop.pos.x, cop.pos.y, 1, 1)
        image(PopoIMG, cop.pos.x, cop.pos.y, popoW, popoH)


def keyPressed():
    if key == "r":
        initialize()
        print("New Game")


def mousePressed():
    rPos = PVector(mouseX, mouseY)
    initialize(rPos)
```

RobberLib.py

```
class Robber:
    pos = None
    speed = 0
    col = None

    last = None

    def __init__(self, x, y):
        self.pos = PVector(x, y)
        self.speed = 1
        self.col = color(255, 33, 33)
        self.last = PVector(0, 0)
```

CopLib.py

```
class Cop:
    pos = None
    speed = 0
    col = None

    def __init__(self, x, y):
        self.pos = PVector(x, y)
        self.speed = 1
        self.col = color(33, 198, 255)
```