LIFE OF MARK THE LIST

Indholdsfortegnelse:

Life of Mark - The List - Synopsis
Hodyah, Lac, Mads, Kristian H
06/06-2020

Introduktion:

- Life of Mark er en fiktiv spilserie, som handler om Marks liv som gymnasielærer. Versionen ”The List” er baseret på dengang Mark mistede sin PC i toget, hvilket gjorde at han blev nødt til at bruge en anden computer. Vi fremstiller en absurd fortælling om hvordan Mark skal finde sin PC igen, for at kunne tjekke fravær. Herfra kommer “The List”, hvilket refererer til klasselisten på Marks computer. For at få klasselisten tilbage møder Mark en masse udfordringer som skal takles, og det bliver alt sammen fortalt i en ufatteligt absurd, men samtidig fangende fortælling.

  Denne aflevering indeholder en demo af spillet. Formålet med demoen er at vise de primære funktioner i spillet, og give spilleren en idé om hvad spillet går ud på.

Opgavebeskrivelse:

- Efter et forløb som omhandlede programmeringssproget Python, fik vi til opgave at bruge udvidelsen ”pygame” til at kode et spil. Kravene til spillet var at det skulle afleveres i fire forskellige iterationer, hvor der skulle kunne ses fremskridt hver gang. Til slut skulle spillet vi havde lavet afleveres sammen med en uddybende beskrivelse og manual til spillet.

Spilkoncept:

Spiltype

- Hvis vores spil skulle udgives på Steam, ville vi ”tagge” spillet som værende under følgende kategorier: ”Indie”, ”RPG” og ”Adventure”. Spillet handler nemlig i stor grad om at udforske forskellige steder. I det fulde spil ville man starte på skolen

Stilen - Hvordan spillet ser ud, og hvorfor

- Vi vidste fra starten at vi ville lave vores egen stilart og derfor egne assets til spillet. For at gøre det nemmest muligt for os har vi valgt at lave bruge pixel art som design, i stedet for at vælge et andet design som kræver flere detaljer. På den måde kan vi være mere effektive med vores tid og fokusere mere på selve programmeringen af spillet.

  Udover at det er nemmere for os at lave pixelart, synes vi helt sikkert også at pixelart bringer noget nostalgi frem i de fleste som har haft spil som Super Mario og Pokémon i deres barndom.

Styring - hvordan man spiller spillet

Styring af menu:

- Man bruger sin mus til at trykke på menuens knapper

Styring af karakter:

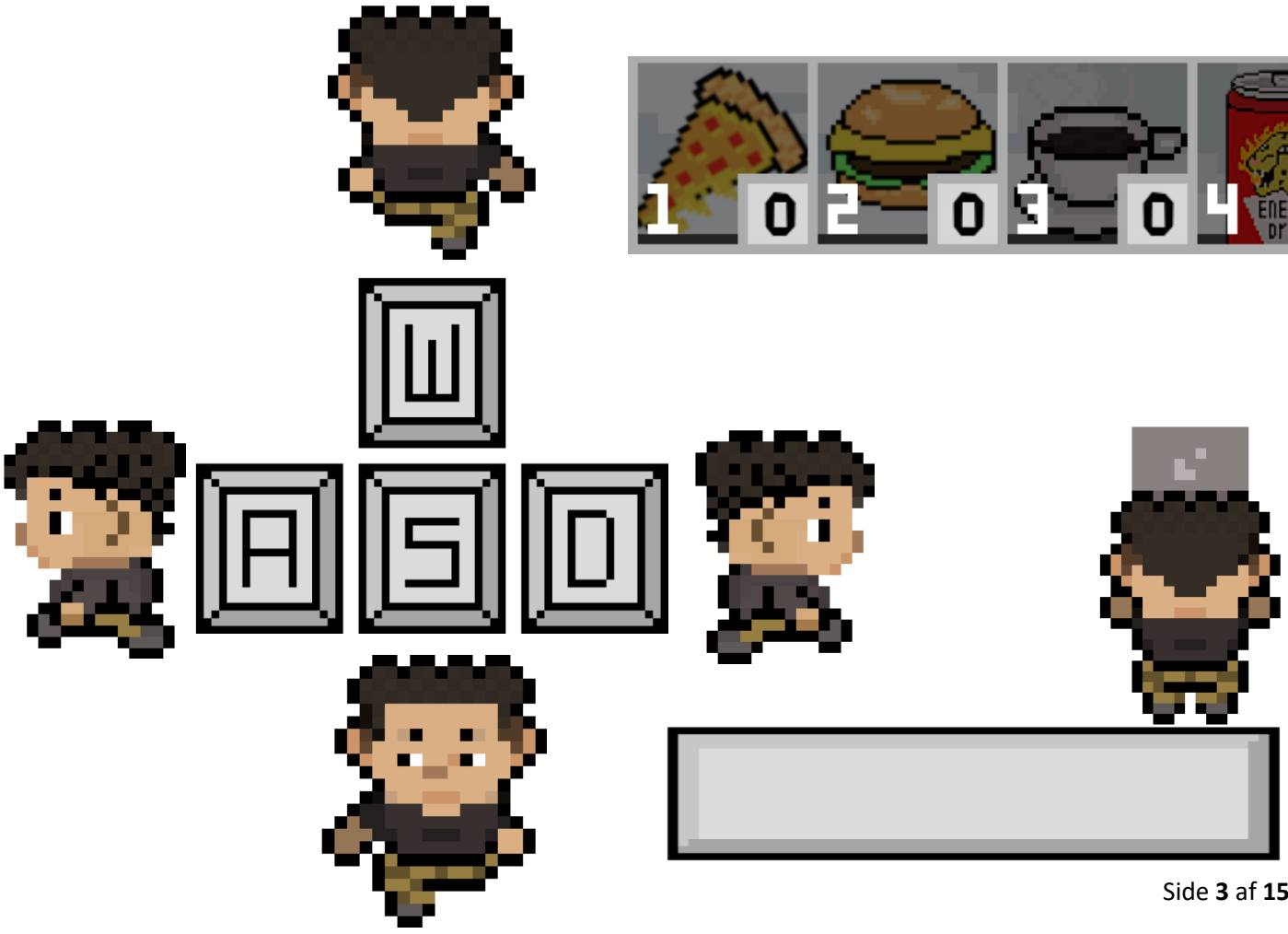
- **W** - Gå frem
- **A** - Gå venstre
- **S** - Gå ned
- **D** - Gå højre

- **Space** - Angrib

- **1** - Spis pizza
- **2** - Spis burger
- **3** - Drik kaffe
- **4** - Drik energidrik

- **L** - gem spillet

- **Esc** - Åben hovedmenu



Life of Mark - The List - Synopsis
Hodyah, Lac, Mads, Kristian H
06/06-2020

Spillets bestanddele

Items:

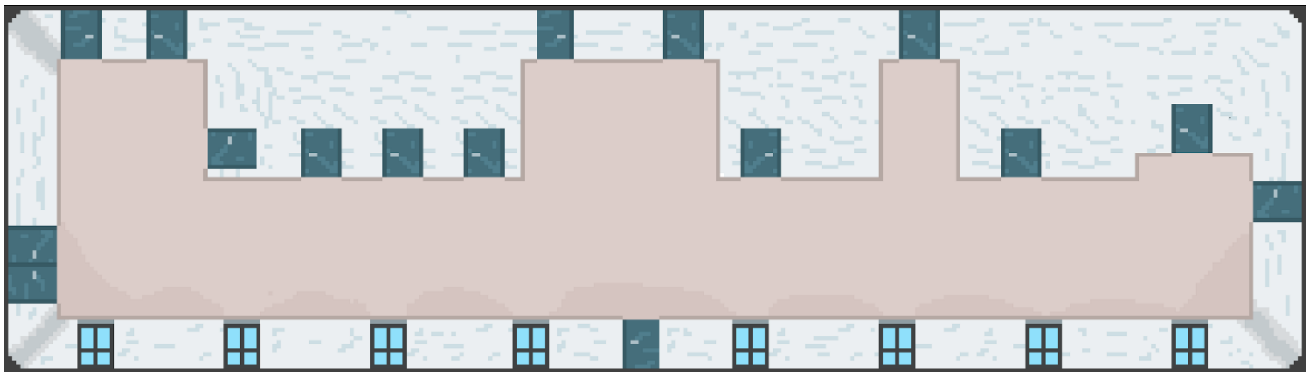
- Items er genstande i spillet som spilleren kan samle op bruge. De forskellige items kan give spilleren mere liv i mængder alt efter hvilket item det er.

Karakterer:

- Karaktererne i spillet er baseret ud fra samme skabelon - Mark-figuren. Her er nogle eksempler:

Baggrunde:

- Baggrundene i demoen er lavet på nogenlunde samme måde, da de alle sammen foregår i samme ”del” af spillet. Her er et eksempel på hvordan baggrundene i demoen ser ud:



Life of Mark - The List - Synopsis
Hodyah, Lac, Mads, Kristian H
06/06-2020

Knapperne i hovedmenuen:

- Knapperne i hovedmenuen skifter farve og siger en lyd når man klikker på dem:

Lydeffekter:

- Lydeffekter gør utroligt meget for et spil. Derfor har vi taget disse forskellige lydeffekter med, for at gøre spillet mere fængende og dybdegående.


BroByggerDamaged.wav


drinkSound.wav


eatSound.wav


failureToEat.wav


PizzaPickup.wav


walksound.wav


selectMenu.wav

Sange:

- Vi benytter musik i baggrunden af spillet og i hovedmenuen. Den første herunder afspilles i hovedmenuen, de tre andre i spillet.


MainMenuMusic.mp3


EmotionalJegGuess.mp3


Violin_Background.mp3


kindahipandoldsong.mp3

Målet med spillet - historien

- I historien er det spilleren som er og styrer hovedkarakteren, Mark. Det betyder at spilleren træffer alle de valg og beslutninger som Mark kan tage. Hele historien er inddelt i 12 scener, hvorfra den første scene er vores demo. I hver scene er der en cut-scene, hvilket er en for programmeret video i spillet, de enkelte cut-scener er der hvor historien blive fortalt.

- Hvordan vinder man?
  Selve spillet kommer til at være et storyspil. Så den måde man vinder spillet på, er ved at komme til enden af spillet Dvs. At spilleren skal gennemføre spillet samt de tilhørende missioner undervejs

- Hvordan taber man?
  Der ville være nogle udfordringer i løbet af spillet og hvis man ikke kan klare udfordringerne, kommer man tilbage til sidste gem dvs., at man dør. Det kan også være at du rage-quitter hvilket betyder du har tabt til et forprogrammeret program. Så kan vi godt forstå, hvis man bliver lidt sur.

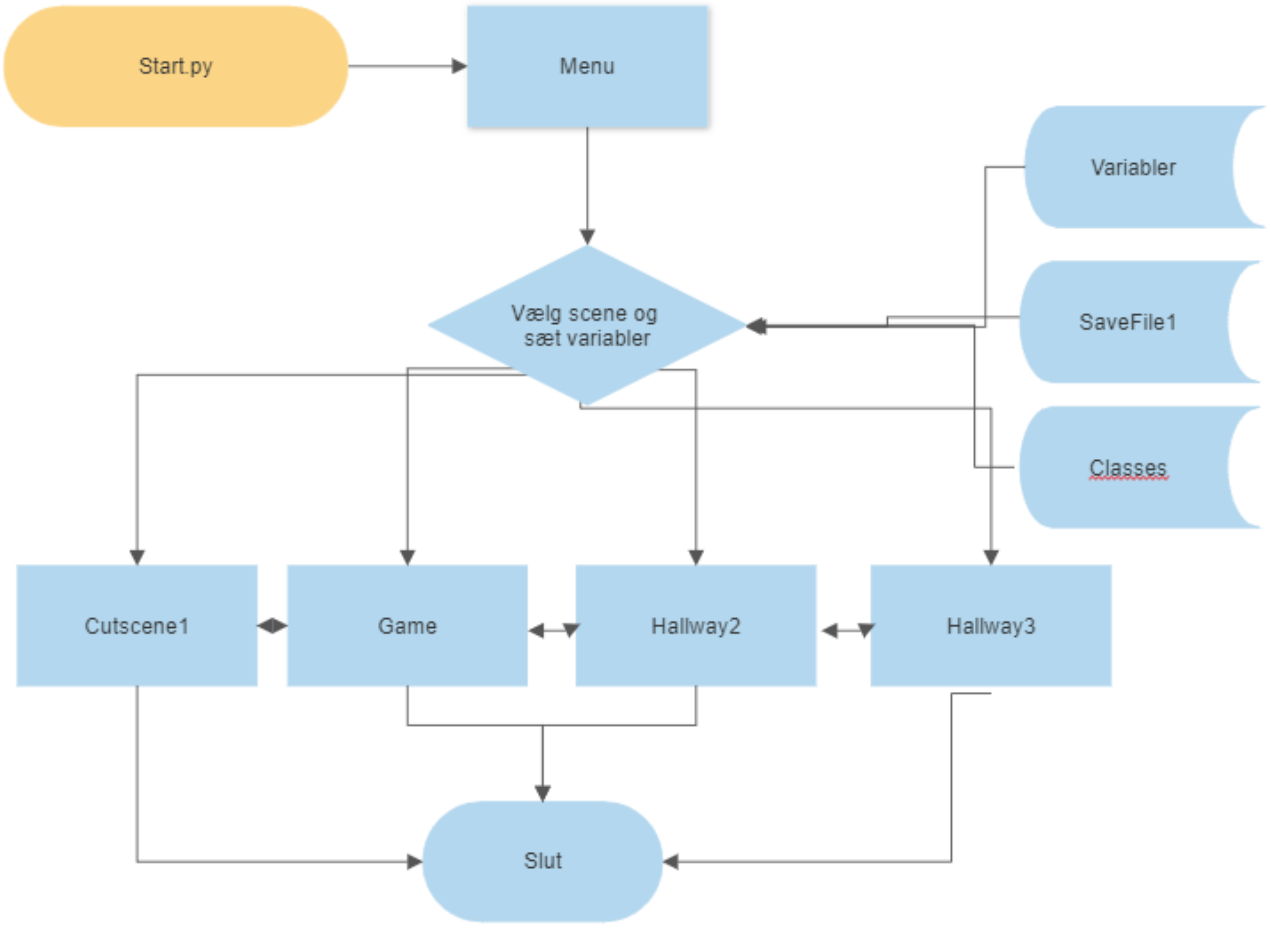
Kode:

Hvordan er koden opbygget?

- Vores kode er opdelt i forskellige scripts, til forskellige scener. Derudover har vi en masse scripts som fungerer som datasæt og instruktioner. På vores flowcharts som er indsat herunder, ses det hvordan de forskellige scripts hænger sammen, og derefter dykker vi dybere ned i nogle af dem.

Flowcharts

Struktur over hele programmet:

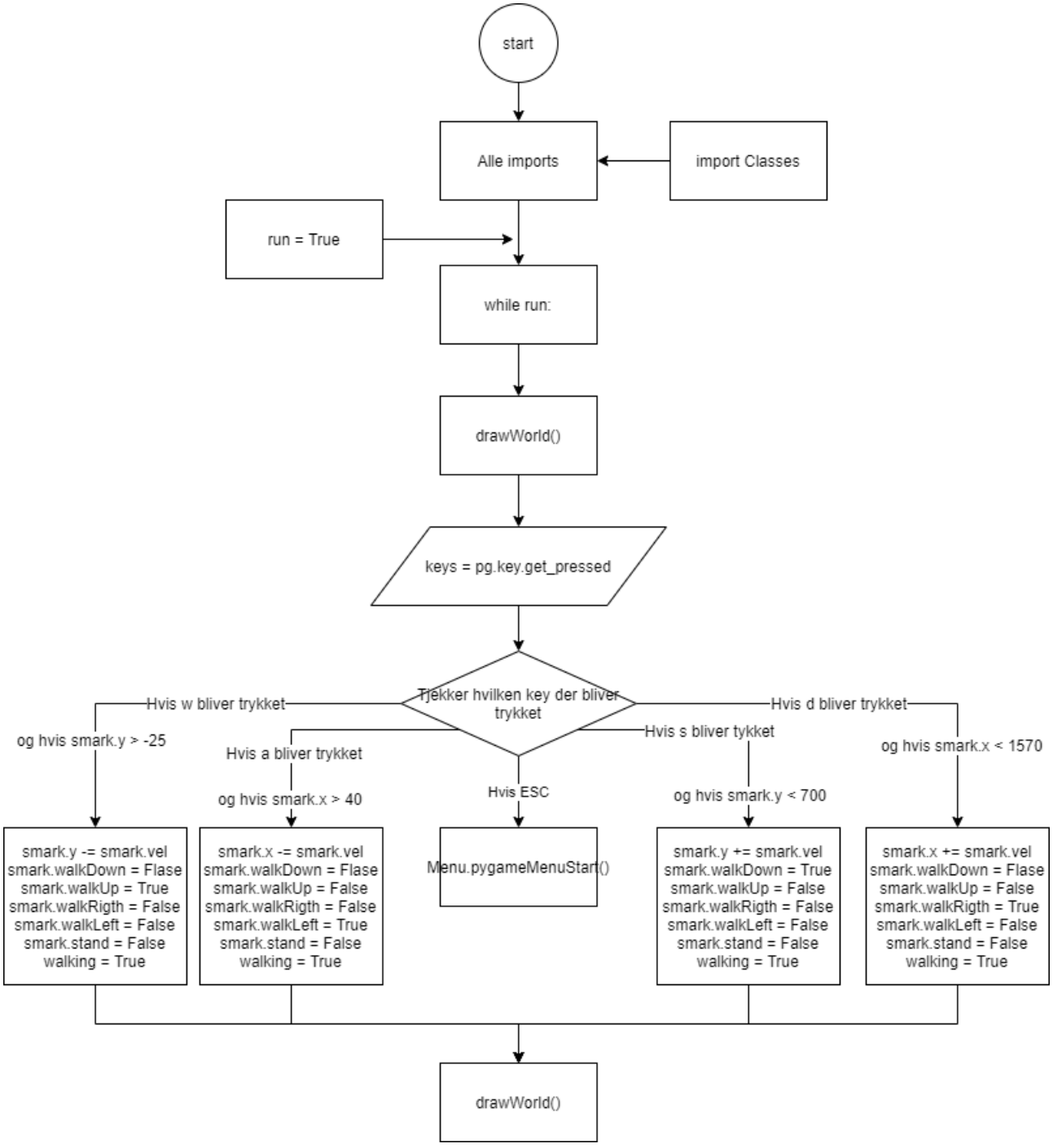


Menu:


Start
Alle imports
import classes
run = True
while run:
drawWorld(baggrundValg)
mx, my = pygame.mouse.get_pos
Tjekker hvilken knap mx, my er over
mx, my er over start
mx, my er over setting
mx, my er over quit
"Tegner" to scrolls omkring start
"Tegner" to scrolls omkring settings
"Tegner" to scrolls omkring quit
Du klikker på mouse1
Tjekker hvilken knap mx, my er over
mx, my er over start
mx, my er over setting
mx, my er over quit
Game.start()
startSettings()
run = False
pg.quit()

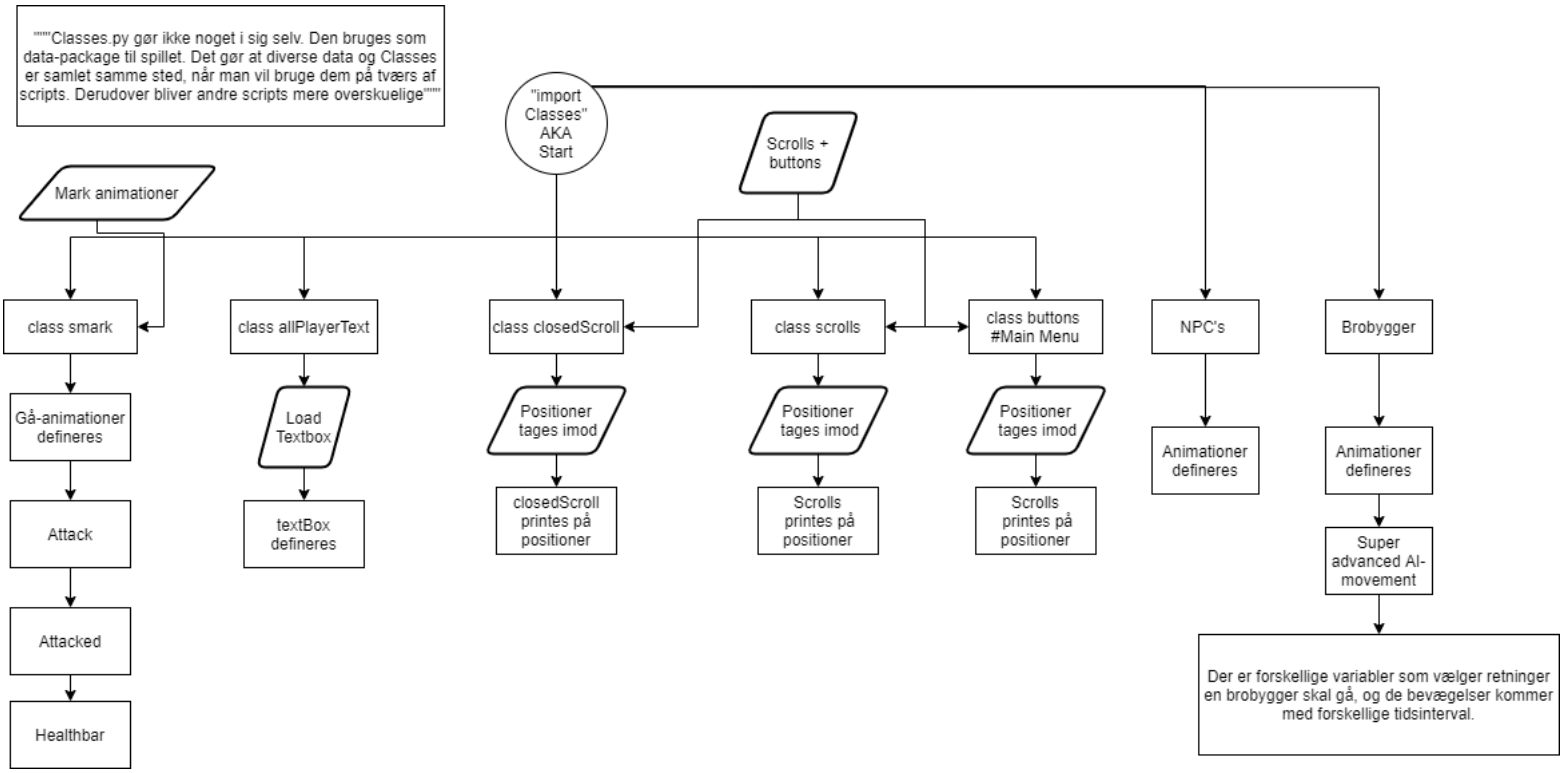
Forskellige scener:

- De forskellige scener er opbygget ud fra samme princip som ses herunder. Dog er der en del variation imellem de forskellige scener, afhængig af de forskellige ting som er i scenen (brobyggere, borde, kollision, osv.)

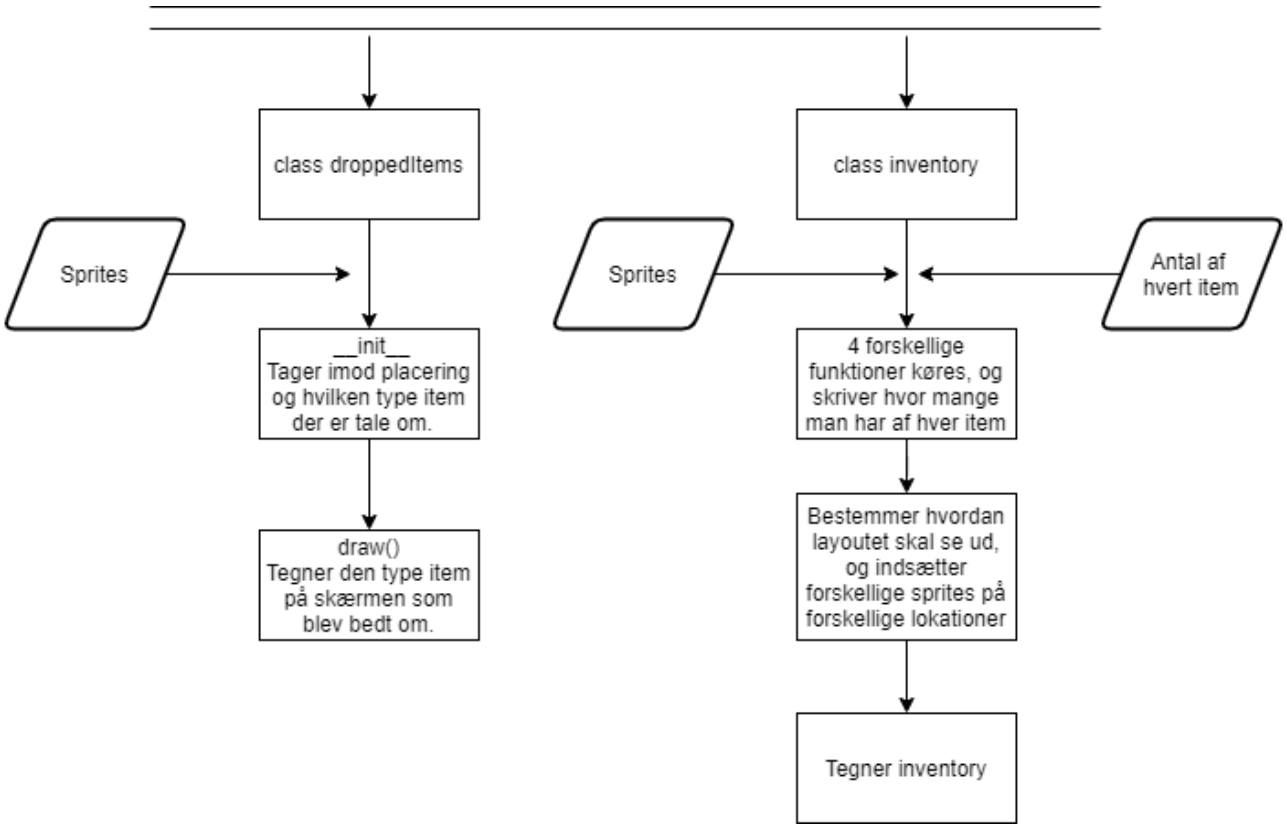


Life of Mark - The List - Synopsis
Hodyah, Lac, Mads, Kristian H
06/06-2020

Classes - Flowchart 1:



Classes - Flowchart 2 (samme script, lavet I 2 forskellige flowcharts, for at skabe mere plads til billederne):



Eksempler på datatyper og lignende (f.eks. typekonvertering, funktioner, objekter og classes)

Variabler:

```
fps = 60
```

Float:

```
tick = pg.time.get_ticks() / 1000
```

Int:

```
health = 1000
```

String:

```
            f.write("Variabler.energidrik = " + str(Variabler.energidrik) + "\n")
```

if, elif og else

```
if self.movementVar <= 5:
            pass
        elif self.movementVar < 10 and self.movementVar > 5:
            pass
        elif self.movementVar < 30 and self.movementVar > 20:
            pass
        elif self.movementVar > 30:
            self.movementVar = 0
        else:
            pass
```

Try/except:

```
        try:
            pizza1.draw(win)
        except:
            pass
```

While loop:

```
while run:
```

For loop:

```
        for event in pg.event.get():
```

```
            if event.type == pg.QUIT or keys[pg.K_ESCAPE]:
                Menu.pygameMenuStart()
```

Funktioner:

```
def start():
    import Menu
    import Game
    walkAllowed_A = True
    walkAllowed_S = True
    walkAllowed_D = True
    walkAllowed_W = True
    eatingAllowed = False
    tick = 0
    broByggerCoolDown = 0
    pg.mixer.music.set_volume(0.07)
```

Class som tager imod inputs:

```
#Player
class smark(object):
    def  __init__(self, x, y,):
```

Eksempler på specifikke funktioner og classes i koden, og hvad de gør

Healthbar:

- Virker ved at tegne forskellige rektangler ovenpå hinanden. Et rødt rektangel sættes bag et grønt, og den grønnes længde varierer efter hvor mange liv man har som spiller. (healthBarFront)

```
    def healthBar(self):
        healthBarBack = (50, 1000, 250, 40)
        healthBarFront = (50, 1000, Variabler.health/4, 40)
        healthBarOutline = (50, 1000, 250, 40)
        pg.draw.rect(win, (255,0,0), healthBarBack, 0)
        pg.draw.rect(win, (0,255,0), healthBarFront, 0)
        pg.draw.rect(win, (0,255,0), healthBarFront, 2)
        pg.draw.rect(win, (0,0,0), healthBarOutline, 3)
```

Animationer som køres for spilleren:

- Animationerne for spilleren er defineret som en liste af billeder. Variablen ”walkCount” tæller hver gang der går en frame. Billedet som vises i animationen, er så defineret som walkCount divideret

med 3. Det vil for eksempel sige at hvis walkCount er 12, så vil plads nummer 4 i listen blive valgt. DSesuden bliver healthbar() kørt, for at opdatere hvis der skulle være sket en ændring.

```
    def draw(self, win):
        if self.walkCount + 1 >= 27:
            self.walkCount = 0

        if not(self.stand):
            if self.walkDown:
                win.blit(markWalkDown[self.walkCount // 3], (self.x, self.y))
                self.walkCount += 1
                self.hitbox = (self.x, self.y, 170, 200)
            elif self.walkUp:
                win.blit(markWalkUp[self.walkCount // 3], (self.x, self.y))
                self.walkCount += 1
                self.hitbox = (self.x, self.y, 170, 200)
            elif self.walkRight:
                win.blit(markWalkRight[self.walkCount // 3], (self.x, self.y))
                self.walkCount += 1
                self.hitbox = (self.x, self.y, 160, 200)
            elif self.walkLeft:
                win.blit(markWalkLeft[self.walkCount // 3], (self.x, self.y))
                self.walkCount += 1
                self.hitbox = (self.x, self.y, 160, 200)
        else:
            if self.walkDown:
                win.blit(markStandDown, (self.x, self.y))
                self.hitbox = (self.x, self.y, 170, 200)
            elif self.walkUp:
                win.blit(markStandUp, (self.x, self.y))
                self.hitbox = (self.x, self.y, 170, 200)
            elif self.walkRight:
                win.blit(markStandRight, (self.x, self.y))
                self.hitbox = (self.x, self.y, 160, 200)
            elif self.walkLeft:
                win.blit(markStandLeft, (self.x, self.y))
                self.hitbox = (self.x, self.y, 160, 200)
            else:
                win.blit(markStand, (self.x, self.y))
        self.healthBar()
```

Inventory-class:

- Inventory klassen har fire forskellige funktioner som definerer tekst, som viser hvor mange man har af hver item. Der er også en funktion, draw(self, win), som kører de fire tidligere funktioner og placerer en skabelon til inventory og indsætter ikoner til hvert item bagved. Hele classen skaber dette resultat, og printer det på skærmen:


```
inventoryBackground = pg.image.load("assets/Inventory/Inventory.png")
inventoryPizza = pg.image.load("assets/Inventory/minipizza.png")
inventoryBurger = pg.image.load("assets/Inventory/miniburger.png")
inventoryKaffe = pg.image.load("assets/Inventory/kaffe.png")
inventoryEnergidrik = pg.image.load("assets/Inventory/energidrik.png")
class inventory(object):
    inventoryX = 1470
    inventoryY = 960

    def pizzaInvCount(self):
        import Tekst
        fontInventory = pg.font.Font("assets/invFont.ttf", 36)
        Text = str(Variabler.pizza)
        textSetting = fontInventory.render(Text, Tekst.run, Tekst.black1)
        textRect = textSetting.get_rect()
        textRect.center = (self.inventoryX+91, self.inventoryY+100)
        win.blit(textSetting, textRect)

    def burgerInvCount(self):
        import Tekst
        fontInventory = pg.font.Font("assets/invFont.ttf", 36)
        Text = str(Variabler.burger)
        textSetting = fontInventory.render(Text, Tekst.run, Tekst.black1)
        textRect = textSetting.get_rect()
        textRect.center = (self.inventoryX+204, self.inventoryY+100)
        win.blit(textSetting, textRect)

    def kaffeInvCount(self):
        import Tekst
        fontInventory = pg.font.Font("assets/invFont.ttf", 36)
        Text = str(Variabler.kaffe)
        textSetting = fontInventory.render(Text, Tekst.run, Tekst.black1)
        textRect = textSetting.get_rect()
        textRect.center = (self.inventoryX+318, self.inventoryY+100)
        win.blit(textSetting, textRect)
```

```
    def energidrikInvCount(self):
        import Tekst
        fontInventory = pg.font.Font("assets/invFont.ttf", 36)
        Text = str(Variabler.energidrik)
        textSetting = fontInventory.render(Text, Tekst.run, Tekst.black1)
        textRect = textSetting.get_rect()
        textRect.center = (self.inventoryX+427, self.inventoryY+100)
        win.blit(textSetting, textRect)

    def draw(self, win):
        win.blit(inventoryPizza, (self.inventoryX-2, self.inventoryY))
        win.blit(inventoryBurger, (self.inventoryX+115, self.inventoryY+13))
        win.blit(inventoryKaffe, (self.inventoryX+180, self.inventoryY-24))
        win.blit(inventoryEnergidrik, (self.inventoryX+291, self.inventoryY-35))
        win.blit(inventoryBackground, (self.inventoryX, self.inventoryY))

        self.pizzaInvCount()
        self.burgerInvCount()
        self.kaffeInvCount()
        self.energidrikInvCount()
```

Save-funktion:

- Forskellige variabler gemmes i saveFile1.py. På den måde kan vi bestemme hvad der skal være gemt når spillet åbnes igen.

```
if keys[pg.K_l]:
            f = open("saveFile1.py", "w")
            f.write("import Classes" + "\n")
            f.write("import Variabler" + "\n")
            f.write("x = " + str(smark.x) + "\n")
            f.write("y = " + str(smark.y) + "\n")
            f.write("smark = Classes.smark(x, y)" + "\n")
            f.write("smark.walkDown = " + str(smark.walkDown) + "\n")
            f.write("smark.walkUp = " + str(smark.walkUp) + "\n")
            f.write("smark.walkRigth = " + str(smark.walkRight) + "\n")
            f.write("smark.walkLeft = " + str(smark.walkLeft) + "\n")
            f.write("smark.stand = " + str(smark.stand) + "\n")
            f.write("walking = " + str(walking) + "\n")
            f.write("scene = " + str(scene) + "\n")
            f.write("Variabler.health = " + str(Variabler.health) + "\n")
            #inventory
            f.write("Variabler.pizza = " + str(Variabler.pizza) + "\n")
            f.write("Variabler.burger = " + str(Variabler.burger) + "\n")
            f.write("Variabler.kaffe = " + str(Variabler.kaffe) + "\n")
            f.write("Variabler.energidrik =" + str(Variabler.energidrik) + "\n")
            f.close()
```

Kilder:

Vi har selv fremstillet de fleste af vores assets, og ellers er de næsten helt sikkert redigeret. Her er de få kilder vi har brugt til assets:

Dinotype, dafont.com, "HoPoYa Studio", besøgt 27/05-2020, https://www.dafont.com/dinotype.font

Failure 1, freesound.org, "Leszek Szary", besøgt 29/05-2020
https://freesound.org/people/Leszek_Szary/sounds/171673/

Coin4.wav, freesound.org, "Free Rush", besøgt 25/05-2020 https://freesound.org/people/Free-Rush/sounds/336933/

Single Water Droplet Sound, soundbible.com, Mike Koenig, besøgt 06/05-2020,
http://soundbible.com/384-Single-Water-Droplet.html

FOOT1.wav, sdltd.com, Ukendt skaber, besøgt 04/05-2020
http://www.sdltd.com/assets/audio/noise_cancellation_tutorials/events/FOOT1.wav

Oversigt- Hele vores kode:

__init__.py - Tomt script - tillader imports

start.py

Tekst.py

Menu.py

Variabler.py

Classes.py

saveFile1.py

cutscene1.py

Game.py (Scene 1)

Hallway2.py (Scene 2)

Hallway3.py (Scene 3)