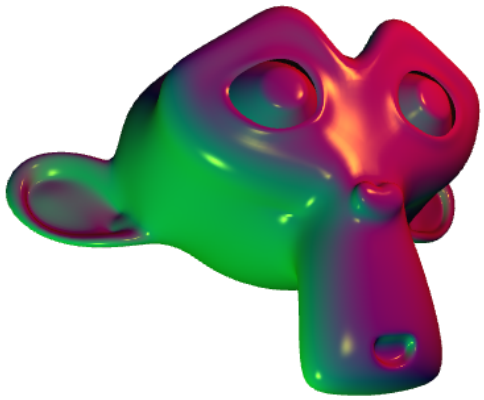
Projektarbejde, Snake

Anna Ølgaard Nielsen
s144437

Christian Søholm Andersen
s103080

Mathias Enggrob Boon
s144484

Van Anh Tri Trinh
s1444449



Introduktion til Softawareteknologi, 02121
Technical University of Denmark

September-December 2014

Preface

HELLO THIS IS PREFACE

1 Struktur af Simpel-Snake

Snake-spillet er lavet efter et MVC-design, hvormed selve spillet, styringen af spillet og den visuelle repræsentation af spillet holdes adskilt i tre dele.

1.1 Styring

I simpel-snake bruges der kun 4 taster til input, nemlig de fire retningstaster, som derfor er defineret i control. Control-klassen har en metode, keyPressed, der kaldes hver gang der tastes. Hvis tasten er en af de fire retningstaster, flyttes slangen i den tilsvarende retning. Hvis ikke, sker der intet.

1.2 Model

Spillets model består af en række klasser, der tilsammen udgør selve spillet. Klassen "Field" bruges til at definere et objekt, som kan opdele banen. Funktionenen equals er defineret i denne klasse, og bruges til at undersøge om to objekter ligger på samme felt, f.eks. slangens hoved og æblet. Æblet er defineret i klassen "Food", hvor datafeltet "position" afgør dets nuværende position. Slangen selv er defineret i klassen "Snake". Slangens krop består af en række felter, hvoraf det første er hovedet, og de resterende er kroppen. Koordinaterne for disse felter er gemt som elementer i en ArrayList kaldet "positions". Det første element er slangens første led, hovedet, andet element er slangens andet led osv. Når et nyt led tilføjes, tilføjes et nyt element til listen. Funktionen createStartingSnake laver slangen ved at bestemme banens centrum, og tilføje to elementer til "positions" med centrumfeltet og feltet til venstre for dette felt.

1.2.1 Slange

ArrayList til at holde slangens punkter. Element 0 er altid hovedet - Sidste element er altid halen Direction = Sidste retning slangen bevægede sig. Når slangen flytter sig, opdateres dens felter "bagfra", dvs. hale bliver til næstsidste position, næstsidste position bliver tredjesidste osv. Sidst flyttes hovedet til den nye position.

1.2.2 View (Brugerflade)

2 Udviklingsproces - Simpel Udgave

2.1 Controller

2.2 Model

2.2.1 Field vs. point til at bestemme koordinater

2.2.2 Opbygning af banen

Til designet af selve banen, som slangen bevæger sig på, forelå to muligheder. Den ene var at lave et to-dimensionelt array af datatypen enum, hvis størrelse afgør banens endelige størrelse.

Et array [10][5] vil f.eks. give en bane med længden 10 og bredden 5. Elementerne i arrayet kan da f.eks. være et blankt felt, et æble, et led af slangen osv. Dette gør det nemt at introducere nye spilelementer i fremtiden, f.eks. bonus-point, vægge, miner osv. Spillet kan da nemt visualiseres ved at definere et billede for hvert spilelement. Programmet kan da tegne det tilsvarende objekt på korrekte plads. ULEMPER - observer?

En anden metode er, at lade de forskellige spilelementer være defineret i deres egne klasser, så f.eks. SnakeFood er en klasse for sig selv, SnakePlayer er en klasse for sig selv osv. Hver klasse har de funktioner, der er relevante for dem, f.eks. getPosition for at give deres nuværende position. Programmet tegner da spillet hver gang en tur afsluttes, dvs. når alle elementer som skal ændres, er ændret. Programmet har fået defineret billedet for de forskellige elementer, og modtager deres position vha. en getPosition metode. Ulempen ved denne metode er, at tegning af programmet gøres mere kompliceret. I den første metode er alle felter allerede defineret, og for at ændre dem behøves det blot at ændre værdien. Ønskes et spilelement ændret eller introduceret med den anden metode, skal der laves et nyt element.

Den anden metode blev valgt til spillet,

2.2.3 Banens størrelse

Et vigtigt element i spillet er "æblet" slangen går efter. Idet æblet har en bestemt position, blev en metode lavet, således at æblet får en ny tilfældig position, når slangens hoved når æblet. HVORDAN?!

2.2.4 Snake

Da slangen i snake-spillet består af en række felter, som alle har netop en koordinat i forbindelse med de resterende, er en effektiv måde at bestemme slangens position på en LinkedList, idet denne datastrukturer er fleksibel i størrelse og passer til formålet. I første omgang blev en ArrayList benyttet, men blev erstattet, idet slangens første led (hovedet), altid placeres som element 0. Det er da mere effektivt at benytte en LinkedList, for at undgå at skulle flytte alle elementer, hver gang længden øges. Bevægelse af slangen fungerer er opdelt vha. en række if og else statements. Det undersøges først, om slangen bevæger sig ind i et felt, hvor der er et æble. Hvis dette er sandt, tilføjes der et nyt led til slangens LinkedList på position 0 med samme position som æblet. Dette nye element er da slangens nye hoved. Er der ikke et æble i feltet, undersøges det om et af slangens andre led (undtagen halen) har samme position. Hvis sandt, slutter spillet, idet slangen har ramt sig selv. Hvis ingen af disse kriterier er opfyldt, flyttes slangen normalt i den ønskede retning. Kontrol af retning og bevægelse ud over banen behandles af controller-delen.. Score For at implementere scoren blev der fremlagt to løsninger. Enten at lade scoren være et datafelt i game-klassen, eller at lade det være en klasse for sig selv. Ved at lade scoren være et datafelt, bliver implementationen simplere. At lade scoren være en klasse for sig selv har derimod fordelen, at der kan tilføjes en observer til Score-klassen, som dermed kun opdateres, når scoren ændrer sig. Scoren bliver dermed kun gentegnet, når scoren ændrer sig. Hvis score derimod er et datafelt, tegnes scoren efter hver tur, også selvom scoren er uændret, hvilket er mindre effektivt. Forskellen er dog minimal, og det blev derfor prioriteret at holde scoren som datafelt, hvormed det også bliver simplere at implementere nye score-relaterede funktioner i fremtidige udgaver.

2.3 Brugerflade og visualisering af programmet

2.3.1 Tegning af banen

2.3.2 Vinduestørrelse

Området, som spillet foregår på, skal kunne bestemmes til at være mellem 5x5 og 100x100. Dog er felterne defineret som en brøk af vinduets samlede størrelse, dvs. at f.eks. et felt i en 10x10 bane er 1/10 af vinduets størrelse. Dog kan en stor bane kræve, at selve vinduet også er af en hvis størrelse, så de enkelte felter ikke bliver for små. F.eks. er det svært at spille med en banestørrelse på 75x75, hvis vinduets størrelse er 375x375, idet de enkelte felter da kun bliver 5 pixels brede. For at løse dette problem, kunne der vælges mellem to løsninger. Enten kunne felternes størrelse fastlægges, og vinduets størrelse justeres efter dette. Ønskes det f.eks. at felterne altid har størrelsen 20x20, og at banen skal være 15x25, vil vinduets størrelse blive 300x500. Fordelen ved denne metode er at det sikres, at banen altid er synlig, og at der ikke opstår problemer, fordi forholdet mellem vinduets størrelse og banen størrelse ikke passer sammen. Ulempen ved metoden er, at store baner kan blive for store til at være på en normal skærm, f.eks. vil en bane med felter af størrelsen 20x20 og banestørrelsen 75x75 fylde 1500x1500.

Den anden metode, at gøre vinduet justerbart, løser dette problem, idet vinduets størrelse frit kan justeres som ønsket. Denne metode introducerer dog et andet problem, nemlig at felternes størrelse skal skaleres til at passe vinduet. Nogle opløsninger af vinduet vil ikke være et multiplum af banens størrelse, hvormed elementerne i spillet vil blive aflange. Dette kan dog løses ved at lave en baggrund og låse banens forhold, hvormed banen altid fylder mest muligt af vinduet ud, og den resterende plads bliver udfyldt af baggrunden. For at give brugeren den bedste mulighed for justering blev metode nummer 2 valgt, hvormed banerne altid er synlige til at starte med, og kan justeres efter ønsker derefter.

3 Struktur af endelig version

3.1 Styring

3.2 Model

3.3 Brugerflade

4 Udviklingsproces af avanceret version

4.1 Optegnelse af slange

I simpel-snake kan spilleren se hvor han har været, men ikke i hvilken rækkefølge. Spilkvaliteten kunne dermed forbedres, ved at lade spilleren se, hvordan slangen har bevæget sig. Det ønskede resultat ville være, at hvert af slangens led er forbundet med det forrige og næste, og så har fri plads i de områder, der ikke er forbundet. For at opnå dette blev der lavet en ny klasse, Bodytype, til at definere hvordan slangens led skal se ud. I snake-klassen blev en ny ArrayList tilføjet, nemlig body, som bruges til at bestemme Bodytype for de forskellige led. Bodytype(0) er dermed hovedet, bodytype(1) er første led osv. Hver bodytype består af to datafelter af typen "relation", hvis værdi er afgjort af det forrige og næste led. Er det forrige led f.eks. over det

nuværende, er prev = ABOVE. Er det næste led til højre for det nuværende led, er next = BESIDERIGHT. Leddenes Bodytype skal naturligvis opdateres efter hver tur. Dette gøres af funktionen updateBody, som vha. en for-løkke går gennem alle elementerne. Det undersøges da, om positionen for det forrige og næste led er over, under, til højre eller venstre for det nuværende led, og prev samt next sættes til at passe dette. updateBody kaldes hver gang slangen bevæger sig.

4.2 Lyd

Som en mindre tilføjelse til spillet er lyd tilføjet i form af Audio-klassen. Funktionerne i Audio-klassen bruget til individuelt at spille

4.3 Hovedmenu

4.4 Automatisk bevægelse

4.5 Variatoner af baner

4.6 Multiplayer