

VRIJE UNIVERSITEIT BRUSSEL

COMPUTERSYSTEMEN:

"GROEP 2: PROJECT SNAKE"

Code verslag

Auteurs:

Jérôme BOTOKO EKILA
Arthur CHOMÉ

Richting.:

Faculteit Wetenschappen
Bach. Computerwet.

23 december 2017

Inhoudsopgave

1	Inleiding	1
2	Features	2
2.1	Start menu	2
2.2	Game	2
2.2.1	Slang	2
2.2.2	Appel	3
2.2.3	Muren	3
2.2.4	Score en highscore	3
2.2.5	Pauze knop	3
2.3	Moeilijkheidsgraad	4
2.4	Hulp menu	4
2.5	Exit	4
3	Code	5
3.1	Menu	5
3.2	Game	5
3.3	Apple generator	5
3.4	Wall generator	6
3.5	GUI	6
3.5.1	Wat tekenen?	6
3.5.2	Tekenen van berichten	6
4	Afhankelijkheidsdiagram	7

1 Inleiding

Dit programma werd geschreven in 80386 32-bit protected mode assembly en runt in DOSBox. Voor dit project kozen we om het klassieke arcadespel Snake na te maken. Hierbij controleert de gebruiker een slang die appels eet. Het eten van de appel resulteert in het verhogen van score. De speler verliest indien de slang zichzelf opeet. Naarmate de slang langer wordt, wordt dit moeilijker om te voorkomen. Het eindresultaat van dit project is een werkend Snake-spel. Daarbij werden een aantal extra features toegevoegd om het spel uit te breiden.

2 Features

2.1 Start menu

Wanneer het programma met de x86-emulator wordt gestart, verschijnt het startmenu. In dit menu heeft de gebruiker vier mogelijkheden:

1. Het spel opstarten (play game).
2. De moeilijkheidsgraad aanpassen (difficulties).
3. Naar de hulpsectie gaan (help) waar een kort overzicht staat van de belangrijkste spelelementen en hoe het spel te spelen.
4. Het programma afsluiten (exit) waarbij de gebruiker terugkomt op de DOS-emulator.

Deze opties kunnen geselecteerd worden door het pijltje links te positioneren op de gewilde actie en op 'enter' te drukken. Als alternatief om het programma af te sluiten kan de gebruiker ten alle tijde op 'escape' drukken.



Figuur 1: Kiezen in het menu

2.2 Game

Het doel van dit spel is om een zo hoog mogelijke score te behalen. Het spelscherm is eindig in omvang. Hierdoor eindigt het spel altijd door een botsing van de slang in zichzelf.

2.2.1 Slang

Bij de start van het spel krijgt de gebruiker direct controle van een wit-roze slang. Deze beweegt standaard naar rechts in het begin. De speler verliest indien deze zichzelf opeet of op een groene muur botst. De slang beweegt zich anders naarmate de moeilijkheid van het spel. Hoe hoger de moeilijkheid hoe sneller de slang.



Figuur 2: Het spel

2.2.2 Appel

Een appel verschijnt willekeurig op het scherm en verandert van locatie van tijd tot tijd. Bij het opeten van de appel wordt de slang langer. De gebruiker wordt dan ook beloont met een aantal punten voor het vangen van de appel.

2.2.3 Muren

Om te voorkomen dat het spel lineair aanvoelt zonder verrassingen, worden de muren willekeurig op het spelscherm geplaatst bij elke nieuwe spelsessie. Het aantal muren die gegenereerd worden hangt af van de moeilijkheidsgraad. Zo heeft de moeilijkheid baby geen muren, maar 'insane' een tiental.

2.2.4 Score en highscore

In de linkerbovenhoek zit de score. Deze kan verhoogd worden door een appel op te eten. Naarmate de moeilijkheid daalt krijgt de gebruiker meer punten voor een appel. Zo is een appel in 'baby' dubbel zoveel waard vergeleken met de moeilijkheid 'insane'.

Daarbij wordt de hoogst behaalde score behouden, maar deze wordt wel niet bijgehouden bij het sluiten van het spel.

2.2.5 Pauze knop

Door op 'p'-knop te drukken, kan de speler het spel pauzeren. Hierbij valt het spel volledig stil. De slang beweegt niet en de appel verandert niet meer van positie.

2.3 Moeilijkheidsgraad

Om meer variatie in het spel te krijgen en de vaardigheden van de gebruiker te testen, beschikt de speler de keuze over zes verschillende moeilijkheden. Het verschil tussen een lagere en hogere moeilijkheidsgraad is dat de slang sneller beweegt, het eten van een appel geeft minder punten en er worden meer muren geplaatst op het scherm. Aan de hand van de boven en onder pijlen en 'enter' kan men dit menu navigeren en een nieuwe moeilijkheidsgraad selecteren.



Figuur 3: Kiezen van de moeilijkheidsgraad

2.4 Hulp menu

De voorlaatste keuze in het eerste menu is de help sectie. Hier kan de gebruiker een kort overzicht vinden over de werking van het spel. Om terug naar te keren naar het eerste menu kan de gebruiker op 'escape' drukken.

2.5 Exit

Deze laatste optie sluit het programma volledig af. Als alternatief kan de gebruiker te allen tijde op 'escape' drukken om het programma af te sluiten.

3 Code

De code is onderverdeeld in vier bestanden: `game.asm`, `menu.asm`, `gui.asm` en `rand.asm` (+ de include files). Aan de hand van de Watcom makefile kan de code gecompileerd en gelinkt worden. De resulterende executable heet `'snake.exe'`.

3.1 Menu

De start van het programma bevindt zich in `'menu.asm'`. Dit bestand omvat `'game'` en `'gui'` met de nodige includes. Bij de start wordt de VGA kaart geschakeld naar modus 13h om het tekenen van pixels toe te laten.

We hebben twee verschillende input handlers voor de twee eerste opties. De loop van `'help'` is feitelijk de input handler zelf: wachten op `'escape'`. Ten laatste draait het programma volledig in het spel gedeelte indien game aangeroepen is. De functie `'runGame'` is namelijk een loop die enkel zal terugkeren naar het menu bij het oproepen van exit in `'game.asm'`.

3.2 Game

Voordat men in de interne loops terechtkomt, worden alle spelvariabelen eerst geïnitieerd door `'initGameState'`. Het centraliseren van deze variabelen betekent dat het eenvoudig is om ze snel aan te passen.

In de loop van het spel wordt de interactie procedure aangeroepen en mogelijks op gereageerd (bijvoorbeeld het aanpassen van de volgende richting van de slang). Hierna wordt het spel bevordert met het oproepen van `'updateGame'`. Het spel daarentegen wordt echter enkel echt gevordert om de zoveel iteraties. Dit bepaalt de snelheid van het spel (en vervolgens de moeilijkheidsgraad) en kan gecontroleerd worden aan de hand van de variabele `'GAME_COUNTER_MAX'`. Hierna wordt het spel scherm opnieuw getekend.

Indien de interactie procedure vaststelt dat de `'escape'` knop is ingedrukt, keert het programma in de volgende loop terug naar het menu.

3.3 Apple generator

De appel verandert van tijd tot tijd naar een andere willekeurige positie. De procedures die dit afhandelen zijn `'placeRandomApple'` en `'drawApple'`. `'placeRandomApple'` gaat een nieuwe positie berekenen voor de appel en de counter resetten. Deze laatste houdt bij hoe lang een appel al op dezelfde positie staat. Het berekenen van een nieuwe positie gebeurt door twee keer de procedure `'rand'` op te roepen: de eerste keer voor de x-coördinaat en de tweede keer voor de y-coördinaat. `Rand` steekt in register `'eax'` een 32 bit getal. Dit getal delen we door de breedte (of hoogte) van het scherm. Het resultaat is de rest van de deling (modulo) en een getal tussen 0 en

de breedte van het scherm. Deze coördinaten moeten zich niet in een stuk van de slang of een muur bevinden. Dit is mogelijk door de positie van een appel steeds te vergelijken met elk stuk slang in 'SNAKE_X(/Y)'. Hierbij vermenigvuldigen we de index (ten opzichte van het adres van de kop op positie 0) met 4 bytes aangezien we te werk gaan met double words.

3.4 Wall generator

Muren worden willekeurig gegenereerd met de procedure 'generateWalls' en onderverdeeld in muur segmenten van een bepaalde lengte. De eerste stap in het algoritme is het aanmaken van random x-en y-coördinaten door gebruik te maken van de procedure 'rand' zoals beschreven in het vorige deel.

De richting (verticaal of horizontaal) van het segment wordt bij elk nieuw segment gealterneerd.

3.5 GUI

De slang wordt voorgesteld in twee variabelen: 'SNAKE_X' en 'SNAKE_Y'. Beide zijn ongeïnitieerd gealloceerd in het geheugen. Tupels op dezelfde index van deze twee vormen een stukje van de slang. Het hoofd van de slang is het eerste tupel. Het aantal vereiste tupels is gelijk aan het aantal mogelijke tiles op het spelbord. Dit bedraagt namelijk 1000 (40*25) double words voor beide assen. Hetzelfde principe geldt voor de muren. De grootte van een tile (op de 'grid') wordt gedefinieerd door de GUI.

Op deze manier staat de game los van de GUI. Het spel geeft aan wat er moet getekend worden en geeft 'grid' posities door. De GUI berekent hierna welke pixels in de videobuffer getekend moeten worden.

3.5.1 Wat tekenen?

Om te beginnen kozen we om enkel veranderingen te tekenen op het bord. Dit in tegenstelling tot het bijhouden van wat er zich in elke grid-positie bevindt. Een voorbeeld hiervan is het bewegen van de slang. Eerst tekenen we de staart van de slang weg (kan gemakkelijk berekend worden aan de hand van de variabele 'SNAKE_LENGTH'). Daarna tekenen we een nieuw hoofd op de volgende positie. Natuurlijk moeten we dan wel alle tupels updaten met hun volgende positie. Deze volgende positie is dan het data-element in de vorige 4 bytes (aangezien elk data element 4 bytes groot is) in beide 'SNAKE_X' en 'SNAKE_Y'.

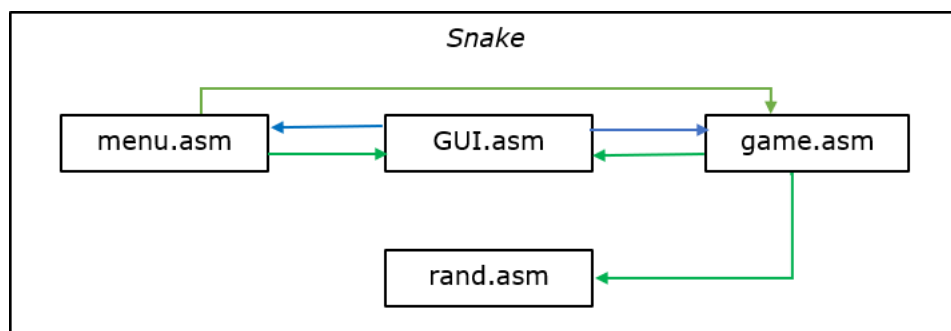
3.5.2 Tekenen van berichten

We kwamen tot de conclusie dat het deconstrueren van een string door elke letter af te gaan en vergelijken niet zo efficiënt verliep. Het tekenen van bijvoorbeeld 'zzz' neemt bijvoorbeeld $26 * 3$ vergelijkingen (voor elke letter



van het alfabet) om te voltooien. Behalve de getallen van de score waren alle andere strings in elke iteratie van het spel steeds hetzelfde. Daarom kozen we om de volgorde van sprites te definiëren in het data segment. Op deze manier kunnen we starten met een pointer naar het eerste data element (de eerste letter) en hierop verder dan itereren.

4 Afhankelijkheidsdiagram

Dit afhankelijkheidsdiagram geeft een geheel beeld over de design van het programma.



Figuur 4: Code design SNAKE

Relatie	Beschrijving
	Roept procedures op van de file waarnaast het wijst. (met <code>.inc</code>)
	Tekent de file waarnaar gewezen wordt