Structuur van computerprogramma's II
Projectverslag
Snake

Bert Van Mieghem
2de Bachelor Computerwetenschappen

Vrije Universiteit Brussel
2016 - 2017

Inleiding

Snake is een simpel 2D spel waarbij je een slang bestuurt met behulp van de pijltjestoetsen. Het doel is om zo veel mogelijk ‘appels’ op te eten waardoor de slang telkens een beetje langer wordt. Hoe langer de slang is, hoe makkelijker hij tegen zichzelf kan botsen. Dit moet je vermijden, anders is het spel over.

Dus eenmaal je slang een indrukwekkende lengte heeft moet je tactisch nadenken over het pad dat je aflegt naar de appel zodat je jezelf niet in de val zet. Ontwijk niet enkel je eigen staart maar zorg ook dat je niet botst tegen een obstakel, dit levert ook game over op.
Gebruik de randen van het scherm om naar de overkant te teleporteren, zo heb je nog een kans om de speciale appel op te eten. Dit is een rode appel die om de zoveel tijd verschijnt, hoe sneller je hem opeet hoe meer punten je zal krijgen. Maar pas op, je krijgt maar 25 “stappen” voordat hij opnieuw verdwijnt.

Ik heb ervoor gekozen om de snelheid van de slang te laten stijgen samen met je score. Dit maakt het spel iets actiever en gewoon leuker naar mijn mening.
Als het vroeg of laat game over is voor jou krijg je de top vijf scores te zien die ooit behaald zijn geweest.

Je kan het spel op elk ogenblik opslaan met 's'. Om het opgeslagen spel te hervatten druk je op 'l', dit kan ook nog nadat het spel is afgesloten en weer opgestart. Met de toets ‘p’ kan je het spel tijdelijk pauzeren en met ‘e’ sluit je het spel, dit kan ook gewoon met het kruisje.

Design

Voor de implementatie van Snake heb ik veel inspiratie uit Minesweeper gehaald. De twee spellen zijn compleet anders maar beide gebruiken een GUI dat een veld toont.

Dynamische geheugenallocatie

De slang en het veld zijn de twee belangrijkste datastructuren van het programma. Het veld is een tweedimensionale array die bij het opstarten van het spel dynamisch gealloceerd wordt door allocate_grid. De array bestaat uit Cell structs, elke cel bevat een status en twee integers (is_apple en is_Sapple) die duidelijk maken of een van de twee soorten appels zich momenteel op de cel bevinden. De status van een cel kan oftewel snake, obstacle of void zijn.

De slang komt uit een gelijkaardige procedure als het veld maar dan voor een eendimensionale array. Allocate_snake vult de array met Snake structs die allemaal een status bevatten samen met een x- en een y-coördinaat die de positie van het slangstukje op het scherm voorstelt.

Main-functie

Het veld en de slang alloceren gebeurt in de main()-functie, hierna wordt de slang op zijn beginpositie gezet, de eerste appel krijgt een plaatsje in het veld en de obstakels worden geplaatst.
Nu wordt de laatste procedure van main() opgeroepen: start_gui. Start_gui gaat het venster tekenen waarmee de interactie met de speler gebeurt. De while-lus in deze procedure zorgt voor het permanent lezen van het toetsenbord en eventueel oproepen van de gameloop.

Gameloop

De gameloop wordt opgeroepen zolang gamestate op 1 staat (dit is altijd, buiten wanneer het spel gepauzeerd is of het scorebord wordt getoond). Elke iteratie zorgt de spel lus ervoor dat de slang beweegt door move_snake op te roepen. Move_snake beweegt het hoofd van de slang in de richting aangegeven door een globale variabele direction (deze wordt aangepast door het indrukken van de pijltjestoetsen) en zal vervolgens elk slangstukje met index 'x' verplaatsen naar de positie van het slangstukje met index 'x - 1'. Ook kijkt move_snake of het hoofd van de slang zich op een cel met een appel bevindt. In dat geval wordt de slang verlengd, de score opgeteld en zal de slang iets sneller bewegen.

De gameloop roept ook de procedure draw_snake op die ervoor zorgt dat elk slangstukje getekend wordt en check_collision, die zal nagaan of het hoofd van de slang tegen een obstakel of tegen zijn eigen staart botst.

De gameloop zorgt eveneens voor het tellen van "stappen" van de slang om te bepalen wanneer de speciale appel moet verschijnen en verdwijnen. Als dit allemaal is gebeurd, wordt de geüpdatete versie van het veld getekend door draw_grid.

Appels

De appel telkens opnieuw laten verschijnen als hij wordt opgegeten gebeurt met de generate_random_apple procedure. Deze neemt één integer argument, een 0 voor een normale appel of een 1 voor een speciale appel. Dit argument wordt in meerdere procedures gebruikt zoals eat_apple en check_apple .

Generate_random_apple gebruikt tweemaal de rand()-functie om willekeurige x- en y-waarden te bekomen. Voordat de appel op deze locatie in het veld wordt geplaatst moet er eerst zeker worden gemaakt dat deze plek geen probleem zal opleveren. Als de appel op een obstakel of op de slang wordt gegenereerd wordt de procedure opnieuw opgeroepen om nog eens te proberen. Enkel als de coördinaten overeenkomen met een cel met status void, wordt de appel effectief geplaatst.

GUI

Al de procedures met betrekking de graphical user interface staan beschreven in GUI.c. Deze maakt gebruik van de SSL- en SSL_TTF library. De inbegrepen toturials werden gebruikt om de meeste procedures te schrijven. Hier wordt onder meer het venster geïnitialiseerd, wordt de keyboard-input gelezen, wordt de score op het scherm getekend, wordt het scorebord getoond, staat de pause/unpause procedure en misschien wel de belangrijkste, draw_grid. Draw_grid gaat elke cel van het veld aflopen en gaat op basis van die zijn status of de is_apple en is_Sapple integers bepalen welke van de vijf afbeelding er moet getekend worden.

Bij game over wordt het venster niet gesloten maar krijg je het scorebord te zien. Dit zijn de top vijf behaalde scores die worden opgeslaan in een '.dat'-bestan. Dit bestand wordt automatisch aangemaakt als het er nog niet is. Het spel herstarten wist deze scores niet.

Om het spel op te slaan worden de belangrijke variabelen zoals de lengte van de slang, de richting, de score, de positie van de appels en meer, extern in een bestand opgeslaan. In dit bestand zal je ook de coördinaten van elk slangstukje vinden. Bij het laden van het spel worden al deze waarden ingelezen en overgeschreven naar je huidig spel. De slang wordt verplaatst en eventueel verlengd of verkort.

Het resultaat van al deze procedures te implementeren, verspreid over vier source files en vier header files. is een werkend snake-spel.

Testen

De errors en waarschuwingen in de terminal waren vaak genoeg om mijn code correct aan te passen maar het implementeren van het project verliep niet altijd even vlot. De grootste hinderpalen waren de allocate_grid en allocate_snake procedures, Het tekenen van de score en het scorebord en het verschijnen/verdwijnen van de speciale appel.

Om het schrijven van deze code zo probleemloos mogelijk te laten verlopen heb ik veel gebruik gemaakt van `printf()` om de staat van variabelen en datastructuren op precieze ogenblikken te kunnen controleren. Online kan je ook zeer hulpzame forums vinden met mensen die precies hetzelfde probleem hadden en als dit nog niet genoeg was had ik de gewoonte om mijn redenering uit te tekenen op papier om precies te kunnen zien waar het fout ging in de code.

Voor het testen van spelfunctionaliteiten was het simpel, speel het spel en zorg dat je zoveel mogelijke situaties naspeelt. Zo had ik bijvoorbeeld snel ontdekt dat als de slang naar rechts gaat en je op links drukt je onmiddellijk sterft, dus zorgde ik ervoor dat dit niet kon. Dit is ook hoe je variabelen zoals de snelheid van de slang, hoe snel de speciale appel verdwijnt, enzovoort het best kan testen en eventueel aanpassen naar gelang.

Om het wegschrijven van data in externe bestanden, het scorebord en het opslaan van het spel dus, te testen moest ik alweer het spel spelen en een score behalen/ het spel opslaan en in het corresponderend bestand de output bekijken. Het schrijven naar een bestand was nooit een groot probleem, maar het inlezen was moeilijker vond ik. Vooral bij het laden van een spelinstantie, om de slang te verplaatsen naar de coördinaten uit het bestand, heb ik lang geprutst. Ik heb veel functies zoals `fgets`, `fscanf`, ... gebruikt voordat de slang effectief correct verplaatst en verlengd/verkort wordt.

Code overzicht

Game.c	Deze source file bevat de code voor de functionaliteiten van snake. Het bewegen/groeien van de slang. Appels opeten, teleporteren, obstakels plaatsen, ...
Grid.c	Procedures met betrekking tot het veld: Het dynamisch alloceren en accesor-functies van het veld en de snake, een appel op het veld plaatsen, ...
GUI.c	Hierin zit al de code die graphical interface mogelijk maakt. Het venster wordt hier geïnitialieerd, het veld wordt elke gameloop iteratie getekend, de score en het scorebord worden getekend, ...
Main.c	Deze source file bevat één procedure: de main-functie. Dit is de functie die wordt opgeroepen als het spel wordt gestart.