Radoslav Glinský

Had - dokumentácia

Tento text slúži ako doprovodná dokumentácia k môjmu zápočtovému programu zimného semestru v roku 2006/2007. Program sa volá Had. Bol napísaný v programovacom jazyku Borland Pascal 7.0. Je to vlastne hra, na prvý pohľad je pre užívateľa, ktorý sa rozhodne si ju zahrať, jednoduchá či na pochopenie či na ovládanie. Hráč 4 klávesami ovláda pohybujúceho sa "hada". Cieľom tejto hry je však zbierať "potravu", a tým pádom sa "had" predlžuje, rastie. Vtip je ale v tom, že hráč nesmie naraziť ani do hraníc hracieho poľa, ani do seba samého.

Po spustení hry Had sa na obrazovke vypíše úvodné menu, ktoré slúži na pohodlné nastavenia hry. Medzi ne patrí rýchlosť pohybu hada po obrazovke, veľkosť hracieho poľa, hustota prekážok na ploche, ktoré hráčovi znepríjemnujú pohyb, ako aj voľba prítomnosti "teleportu" v hre. Hra takisto podporuje súperenie dvoch ľudských protihráľov, čo sa taktiež nastavuje v uvítacom menu. Samotnú hru spustíme výberom položky ´nova hra´, naopak opustíme výberom ´koniec´, respektíve klávesou ESC. V spodnej časti nájdeme ešte informáciu, ako sa had ovláda, rok vzniku programu a meno autora. K jednotlivým položkám menu pristupujeme šipkami nahor a nadol (aktuálnu pozíciu ukazuje červená šipka), parametre meníme stiskom klávesy ENTER. Čo sa týka programovej realizácie, tak vykreslenie menu uskutočnuje procedúra menu. Avšak zmenu parametrov (ak bola nejaká zmena vykonaná) zabezpečuje procedúra kurzor, ktorá sa príkazmi ´gotoxy´ presunie na "správne" miesta a tam prepíše hodnoty nastaviteľných parametrov (ak bola stisknutá klávesa ENTER) alebo vykreslí zmieňovanú šipku (ak bola stlačená klávesa šipka nahor alebo nadol) a to tak, že miesta pod a nad aktuálnou pozíciou začierni (write(´ ´)) a na dané miesto ju pripíše. Treba priznať, že procedúra kurzor prepisuje aj hodnoty, ktoré neboli zmenené, čo je zbytočné. Avšak jedná sa len o zopár jednoduchých príkazov, takže dozaista to nemá žiaden vplyv na chod celého programu. Takáto voľba je aj jednoducho realizovateľná, čiže som sa vyhol zdĺhavému rozvetvovaniu programu.

Prejdime teda na procedúru nova_hra. Tá na začiatku nastaví vstupné data, ako sú skore či počet životov, a podľa v menu zvolenej rýchlosti nastavi "korektnú" rýchlosť=počet milisekúnd, ktorá sa použije v príkaze delay(rychlost), čo zabezpečuje celkové tempo hry, tj. rýchlosť pohybu hada po obrazovke. V tele procedúry nova_hra sa nachádzajú dva repeat until cykly, jeden vnorený v druhom. Ten vnútorný sa opakuje do tej doby, pokým had nenarazí, alebo sa užívateľ rozhodne skončiť hru klávesou ESC(to zámerne ale automaticky ukončí aj ten vonkajší cyklus, a ocitneme sa znovu v hlavnom menu). Rola vonkajšieho repeat until cyklu sa trochu líši v závislosti od počtu hráčov:

1) Hra pre jedného hráča:

Na začiatku vypisuje koľko má hráč životov a nastaví niekoľko hodnot pre druhé funkcie a procedúry. Vyvolá procedúru ploxa, ktorej cieľom je nakresliť jednak hranice hracej plochy, ako aj teleporty(ak boli užívateľom aktivované) a prekážky(takisto len ak boli zvolené). Na teleporty a prekážky som použil príkaz random, ktorý vygeneroval súradnice teleportu resp. prekážok. Pri generovaní teleportu som ošetril, aby oba teleporty neboli na rovnakej polovičke. Pri generovaní prekážok som bral zreteľ aj na to, aby neboli príliľ blízko teleportov. Vlastne všetky tieto kreslenia na obrazovku som uskutočnil pomocou kombinácie príkazov gotoxy(súradnica1,súradnica2), textbackground(číslofarby) a write(´ ´). Doposiaľ je to rovnaké ako pre hru pre 2 hráčov. Pre hru 1 hráča platí: po prebehnutí vnútorného repeat until cyklu testuje, či hráčovi už nedošli všetky životy(implicitne som nastavil počet životov na 3) alebo či hráč vyhral(tj. dosiahol dĺžku rovnú premennej maxdlzka=maximálna prípustná dĺžka hada- nastavená na 30). Ak nastane jedna z týchto možností, tak sa cyklus ukončí a presunieme sa opäť do hlavného menu. Ak hráč vyhral, tak vypíše na obrazovku oznamujúci text.

2) Hra pre 2 hráčov:

Začiatok je rovnaký ako pre 1 hráča, ale líši sa po prebehnuti vnút. repeat until cyklu. V prípade, že jeden z hráčov dosiahol max. dĺžku, tak vyhral, program

to vypíše. Tu by som ešte chcel upozorniť na to, ako hra pre dvoch funguje. Tentokrát sa netestuje, či hráčom došli životy, ale vždy ak jeden z nich narazí(alebo dokonca obaja naraz), tak sa program spýta, či hráči chcú pokračovať v hre vo dvojici. Ľubovoľná klávesa znamená áno, ´n´ alebo ´N´ znamená koniec. Tu by som azda spomenul aj to, že v tomto momente sa hráčom objaví aj správa o tom, ktorý z nich narazil, tj."umrel". Táto správa bude takej farby, ako zomretý hráč(1. Hráč je zelený, 2. hráč je modrý). Ak si uz hráči neprajú ďalej hrať (stisknutím ´n´ alebo ´N´), tak program vypíše hráča, ktorý vyhral, a nahraté skore oboch hráčov (Skore sa hračovi pripočítava s každou "zjedenou potravou", odpočítava vtedy, ked hráč narazí). Opäť program čaká na stisk klávey pre vstup do hl. menu.

Predtým než pristúpime k samotnému jadru hry, to zn. k vnútornému repeat until cyklu, tak by som najprv popísal datový typ, ktorým reprezentujem našeho "hadíka". Špeciálne som si vytvoril jednorozmerné pole hrac (dlžky 2), ktorého členy sú typu record. Typ record v sebe obsahuje údaje ako počet životov(premenná life), skore(skore), aktuálny smer pohybu hada po obrazovke(smer), boolovskú premennú na zistenie, či had nenarazil(skonc), hadovu dlžku(dlzka),dvojrozmerné pole v ktorom sú uložené informácie o súradnicich výskytu hada na obrazovke(pole), integerová premenná udávajúca index poľa, kde sa nach. chvost, a x-ová a y-ová súradnica hadovej hlavy(a,b), a nakoniec nejaké pomocné polia na manipuláciu so súradnicami hada(pomhad,pomhad2). Táto štruktúra poľa s record-ami mi teraz ľahko dovolí eventuálne pridávať nových hráčov, pretože už nemusím vytváraž nové premenné, stačí mi len zvýšiť počet členov poľa hrac. S takýmto poľom sa dobre narába, kedže pri volaniach procedúr a funkcií postačuje odovzdať ako parameter číslo hada(=index poľa hrac), to zn. že nemusím väčšinu premenných ani procedúr duplikovať.

K spôsobu uchovávania informácií o výskyte hada venujem ešte chvíľku, keďže si myslím, že je to dosť podstatná vec, ktorá sa permanentne využíva takmer vo všetkých procedúrach a funkciách. Najsamprv máme dvojrozmerné pole(hrac[].pole), pričom druhý rozmer má len dva indexy. X-ová súradnica na obrazovke sa takto uloží do hrac[].pole[,1] a y-ová do hrac[].pole[,2]. Ďalej mám k dispozícii premennú index, ktorá mi určuje, kde had "začína"(má tam chvost), čiže spolu s premennou dlzka mi jednoznačne určujú aktuálne súradnice hada. Každým jedným posunutím hada na obrazovke(predchádza tomu vygenerovanie nových súradníc hlavy podľa smeru pohybu), sa mi zapíšu súradnice hlavy do hrac[].pole o jeden index(článok) ďalej, pričom sa mi zároveň posúva aj index(ukazuje na index hrac[].pole, kde sa nach. chvost). Ak sa nachádzam na konci môjho poľa, kde zapisujem nové súradnice hlavy, tak jednoducho ich zapíšem na začiatok. To isté sa deje aj s premennou index(ukazuje na chvost), to znamená, že ak sa nachádza na konci poľa, tak mu v danom cykle priradím jednotku. To sa opakuje stále dookola. Azda jedinú výnimku tvorí situácia, keď had "zje potravu", vtedy sa mi premenná index neinkrementuje.

V následujúcich odstavcoch popíšem ako vyzerá vlastne celý priebeh hry(vnút. repeat until cyklus v procedúre nova_hra). Na začiatku sa vždy otestuje, či bola "zjedená potrava", a to udáva bool premenná zobrata. Ak bola zobratá(if zobrata) tak vygenerujem novú. Na to mi poslúži procedúra kokocina. Procedúra pomocou random vyberie súradnice novej potravy a ak sa nezhodujú so súradnicami teleportu, prekážok a aktuálnych súradníc hada (resp. 2 hadov) tak potravu vykreslí. Okrem tohoto má procedúra aj inú úlohu v prípade, že hrá len jeden hráč, kedy sa rýchlosť pohybu hada mení v závislosti na druhu potravy: potrave "priradí" nejakú rýchlosť a k nej prisluchajúcu farbu.

Ak bola stlačená niektorá z kláves, ktorá ovláda hada, tak sa nastaví nový smer pohybu. Predtým si však zapamätám pôvodný smer. To mi poslúži na to, aby som vedel ľahko určiť, či je nový smer presne opačný k tomu pôvodnému.
Ak áno(abs(pov.smer-novysmer)=2, kedže som si smery očisloval od 1 do 4 v smere hod. ručiciek), tak zavolám procedúru vymenhada. Jediná vec, načo slúži vymenhada je to, že obráti všetky súradnice hada(uložené v "hrac[1/2].pole[x,y]"). To značí, že tam, kde bol chvost bude hlava a naopak. Okrem toho ešte procedúra zistí nový

smer pohybu hada a to tak, že si najprv vezme posledné dve súradnice ešte predtým než sa had natrvalo vymení a podľa nich určí nový smer pohybu. Tu ešte upozorňujem, že procedúra sa neuskutoční a had sa nevymení, ak sa had "nachádza v teleporte". To, či sa tam nachádza zistím funkciou je_v_teleporte. Funkcia je_v_teleporte vráti true, ak sa had niektorým svojím článkom nachádza na súradniciach teleportu.

V prípade, že sa súradnice hlavy(hrac[].a, hrac[].b) ocitnú na súradniciach niektorého z teleportov(čiže sa rovnajú), tak sa ihneď vymenia súradnice hlavy za súradnice toho druhého teleportu. Takouto fintou docielim zaujímavého efektu, ako by sa had "skutočne teleportoval".

V dalšom kroku nášho repeat until cyklu sa na základe smeru pohybu zistia nové súradnice hlavy. Tu v podstate ide len o zmenu(+1/-1) jednej súradnice (obidve sa naraz nemôžu zmeniť). Po nadobudnutí nových súradníc hlavy hada si tieto súradnice zapíšeme(do hrac[ ].pole[,]).

Nové súradnice hlavy sa teraz budú testovať, či náhodou had nenarazí. K tomu nám pomôže procedúra inspekcia. V jej tele sa najprv otestuje, či had nenarazil sám do seba. Neskôr či sa hlava náhodou nenachádza na hraniciach hracieho poľa. Potom sa ešte skontrolujú všetky prekážky. Pre 2 hráčov sa ešte skontroluje či jeden nenarazil do toho druhého. V takomto položení sa teraz vypíše hlásenie, že jeden z hráčov umrel(vo farbe umretého hráča), jemu sa aj strhnú body zo skore.

Hned za inspekciou budeme zisťovať, či náhodou sa súradnice hlavy nezhodujú so súradnicami potravy, a to procedúrou test_na_kokocinu. V kladnom prípade prevedie tieto kroky:
1)nastavi bool premennú zobrata na true(aby sa v ďalšom prechode cyklom vygenerovala potrava),
2)zvýši hráčovi, ktorého hlava "pohltila potravu" skore, pričom rozoznáva potravu podľa jej typu(farby a rýchlosti) a na základe toho potom prideľuje skore,
3)zvýši danému hráčovi dĺžku.

A nakoniec azda to najdôležitejšie, čo nám umožňuje aby sme vôbec videli, čo sa to deje v priebehu programu. Vykresľovanie hada má na starosti procedúra kresli. Spracoval som ju tak, aby vlastne nekreslila vždy celého hada, ale stačí nakresliť danou farbou len to miesto, na ktoré ukazujú súradnice novej hlavy(hrac[].a, hrac[].b). Ak by sme to však nechali len takto, had by sa nam stále len predlžoval, aj napriek tomu, že "nezjedol potravu". Preto musíme vždy s nakreslením novej hlavy zároveň aj zrušiť starý chvost, čo neznamená nič iného, ako sa presunúť na súradnice starého chvosta a začierniť ho(príkazmi gotoxy(); textbackground(black); write(´ ´);). Na konci procedúry, ako aj na rôznych iných miestach programu, som použil trik na to, aby sa mi po nakreslení(resp. napísaní nejakého textu) kurzor(viditeľný ako podtržník) nezobrazoval. To som jednoducho realizoval tak, že som sa presunul niekde "do rohu" obrazovky, kde som dal vykresliť čiernou farbou(textcolor(1); write(´ ´,#8);).

Na samom konci cyklu je ešte príkaz delay(rychlost), ktorý udáva tempo hry. Premenná rychlost má v sebe informáciu o počtu milisekúnd a môže sa, ako som už spomínal, meniť a to na základe nastavení v hlavnom menu, alebo v prípade jedného hráča aký typ potravy bol vygenerovaný.

Na záver tejto dokumentácie by som len chcel podotknúť, že na "Hadovi" sa ešte budem snažiť popracovať, doladiť nejaké detaily, niektoré časti programu možno úplne prepísať, ak sa mi neskôr bude zdať, žeby to išlo aj lepšie. A samozrejme vylepšiť hru o nejaké užívateľsky zaujímavé doplnky.