Flocking simulation

Avtorja: Luka Prijatelj in Gašper Kolar

Motivacija: Za temo seminarske naloge sva se odločila, da izbereva problem, ki je pogost v naravi. Želela sva si tudi izbrati problem, kjer bi lahko podatke prikazala v čim bolj grafičnem načinu. Izbrala sva si »*Flocking simulation*« oziroma simulacija jate. To je simulacija jate ptic, ki se morajo med seboj pogovarjati, zato da ne pride do trkov. Te ptice se morajo tudi čim bolj držati svoje jate in obdržati njihovo smer. Zamislila sva si, da računanje smeri in premikanje ptic paralleliziravа z algoritmi in arhitekturami, ki jih bomo spoznali pri predmetu. Parallelizacijo bi naredila v jeziku C++, ki bi vsak premik ptic zapisal v svojo tekstovno datoteko. Za grafično izrisovanje pa bi naredila manjši C# program, ki bi te tekstovne datoteke prebral in zgradil video simulacijo.

Kratek opis: *Flocking simulation* oziroma simulacija jate ima 3 preprosta pravila. Ta pravila so ločenost (angl. Separation), usmerjenost (angl. Allignment) in povezanost (angl. Cohesion). Pravilo ločenosti izbere najbližje sosednje ptice za določenem obsegu. Pravilo usmerjenosti poskrbi, da trenutno izbrana ptica gleda leti v isto smer kot njene sosedne ptice. Pravilo povezanosti pa poskrbi, da se ptice držijo v jati in ne odletijo vsaka v svojo smer. Vsaka ptica ima 4 komponente - vektor pozicije (X in Y koordinati) ter vektor smeri, ki je prav tako vektor hitrosti (X in Y koordinati).

Groba psevdokoda algoritma:

```
N = 200; // stevilo vseh ptic
trenutno_stanje = ptica[N]; // ptica je struktura, ki predstavlja usmerjen vektor
naslednje_stanje = ptica[N];

inicializiraj(trenutno_stanje);

while (1):
	for (i = 0 ... N):
		// poisci_lokalne_ptice vrne vse ptice znotraj nekaga radija podane ptice
		lokalne_ptice = poisci_lokalne_ptice(trenutno_stanje[i], trenutno_stanje);
		//O(N)
		// vrne novo stanje za podano ptico glede na podane lokalne ptice ter 3
		pravila letenja
		naslednje_stanje[i] = novo_stanje(trenutno_stanje[i], lokalne_ptice);
		// O(M) kjer je M stevilo lokalnih ptic

	izrisi_stanje(naslednje_stanje);
	// kjer izrisi_stanje izrise graficen prikaz ptic za podano stanje
	trenutno_stanje = naslednje_stanje;
```

Časovna zahtevnost: Časovna zahtevnost je $O(n^2)$, kjer je *n* število vseh ptic.

Prostorska zahtevnost: Je zahtevnost O(n), kjer je *n* število ptic.

Reference:

[1]: http://www.cse.buffalo.edu/faculty/miller/Courses/CSE633/Cosgrove-Spring-2014-CSE633.pdf

[2]: https://gamedevelopment.tutsplus.com/tutorials/3-simple-rules-of-flocking-behaviors-alignment-cohesion-and-separation--gamedev-3444

[3]: http://www.red3d.com/cwr/boids/

[4]: http://www.red3d.com/cwr/papers/1987/SIGGRAPH87.pdf