Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gašper Kolar, Luka Prijatelj

Simulacija jate ptic

končno poročilo

poraZdeljeni sistemi
univerzitetni študijski program
prve stopnje
računalništvo in informatika

Ljubljana, 2017

Kazalo

Povzetek

Naslov: Simulacija jate ptic

Avtor: Gašper Kolar, Luka Prijatelj

V vzorcu je predstavljen postopek priprave diplomskega dela z uporabo okolja LaTeX. Vaš povzetek mora sicer vsebovati približno 100 besed, ta tukaj je odločno prekratek.

Ključne besede: simulacija, jata, ptice, serijski, paralelni, pThreads, OpenMP, OpenCL, MPI.

Poglavje 1

Uvod

Za temo seminarske naloge pri predmetu smo si izbrali simulacijo jate ptic(ang. Flockin simulation).

Vsako iteracijo je potrebno izračunati novo pozicijo za vsako ptico glede na bližnje ptice. Nova pozicija se izračuna glede na 3 preprosta pravila. Ta pravila so ločenost (angl. Separation), usmerjenost (angl. Allignment) in povezanost (angl. Cohesion). Pravilo ločenosti skrbi, da ptice ne letijo pre blizu druga drugi. Pravilo usmerjenosti poskrbi, da ptice v jati letijo v isto smer. Pravilo povezanosti pa poskrbi, da se ptice držijo v jati in ne odletijo vsaka v svojo smer. Stanje vsake ptice je opisano z štirimi komponentami - vektor pozicije (X in Y koordinati) ter vektor smeri oz. hitrosti (X in Y koordinati).

Iz grobe psevdokode 1 algortma je razvidno, da je osnovna serijska implementacija zelo enostavna. Vsako iteracijo se, glede na trenutno stanje, izračuna novo stanje. Novo stanje za vsako ptico se izračuna le glede na lokalne ptice. To so ptice, ki so znotraj določenega radija oddaljenosti od ptice za katero računamo novo stanje. Iskanje lokalnih ptic ima $O(N)$ časovno zahtevnost kjer je N enak številu vseh ptic v jati. Računanje novega stanje za določeno ptico pa ima časovno zahtevnost enako $O(M)$. Tu je M enak številu lokalnih ptic, ki pa ni nikoli večje od N tako, da je časovna zahtevnost računanja novega stanja za določeno ptico prav tako $O(N)$. Novo stanje

pa je potrebno izračunati za vse ptice v jati tako, da je časovna zahtevnost celotnega problema enaka $O(N^2)$.

Iz psevdokode 1 se prav tako vidi, da v pomnilniku hranimo le dve tabeli ptic ki jih vsako iteracijo zamenjamo. Prostorska zahtevnost algorima

Algorithm 1 Groba psevdo koda serijskega algoritma

```
1: N ← Stevilo ptic
2: trenutnoStanje ← ptice[N]
3: naslendjeStanje ← ptice[N]
4: loop
5:     for i ← 0; i < N; i++ do
6:         lokalnePtice ← poisciLokalnePtice(trenutnoStanje[i], trenutnoStanje);
7:         naslendjeStanje[i] ← novoStanje(trenutnoStanje[i], lokalnePtice);
8:     izrisiStanje(naslendjeStanje);
9:     trenutnoStanje[i] ← naslendjeStanje;
```

Poglavje 2

Serijska implementacija

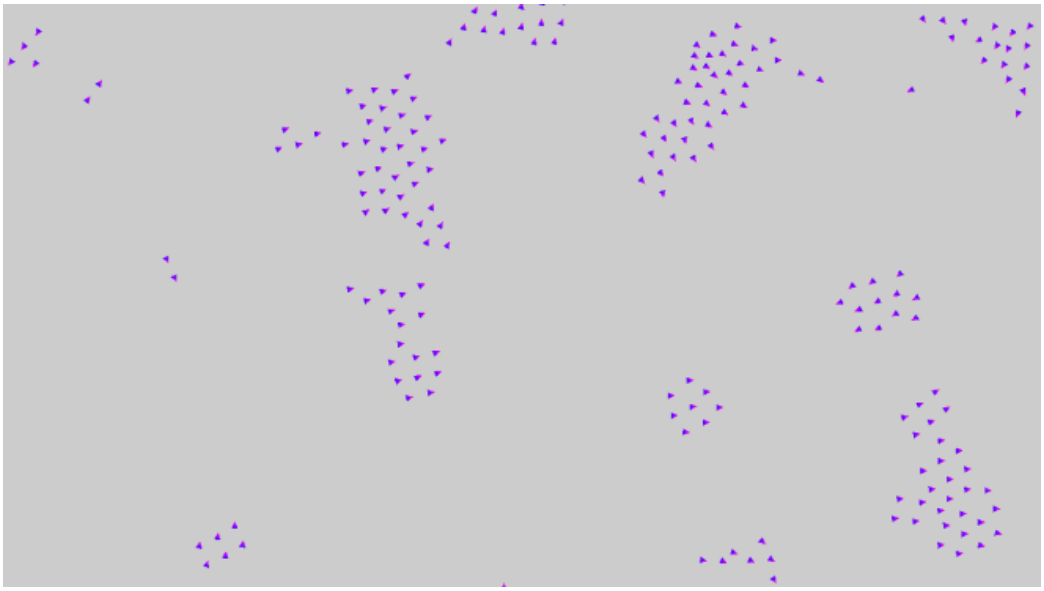
2.1 Osnovna implementacija

Razvoj osnovne implementacije ni predstavljal večjih izzivov. Za razvoj smo uporabili Microsoftov Visual Studio 2015 integrirano razvojno okolje, ki se je izkazalo za zelo koristno, posebno pri odpravljanju hroščev. Nekoliko težav smo imeli le z grafičnim izrisovanjem jate ptic. Saj smo se avtorji tokrat privič srečali z grafično knjižnico OpenGL. Poleg OpenGL smo uporabili še GLFW knižnjico , ki poskrbi za kreiranje grafičnega okna. Končen rezulatat je prikazan na Sliki 2.1. Po približno 10 sekundah leta so se ptice, iz popolnoma naključnih pozicij združile v jate.

Glavna podatkovna struktura programa je *ptica*. *ptica* zajema vse podatke ki so potrebni za opis ene ptice v prostoru. To sta torej dva dvodimenzionalna vektorja *pozicija* in *smer*. Struktura *jata* pa združi vse ptice v tabelo *ptic*.

Serijska implementacija potrebuje kot vhodni podatek le število ptic. Glede na število ptic se ustvari ustrezno velika *jata*.

Ker rezultat vsake iteracije tudi izrišemo na ekran smo se oločili, da bo glavna merska enota število okvirjev na sekundo (ang. Frames per second). Rezulati meritev so tabelirani v Tabela 2.1. Ker smo bili nad rezultati nekoliko razočarani smo se odločili algoritem pohitriti z mrežo.



Slika 2.1: Slika je bila zajeta po približno 10 sekundah prostega leta. Ptice so se združile v jate.

2.2 Implementacija z mrežo

Zaradi relativno poraznih rezultatov smo se odločili, da serijski algoritem pohitrimo. To smo naredili z implementacijo nove podatkovne strukture mreža (ang. Grid). Iz popravljene psevdokode 2 algorirma je razvidno, da se vsako iteracijo ptice najprej razporedi v mrežo na to pa se izračunajo nova stanja ptic. Mreža je ilustrirana na Sliki 2.2. Velikost celice je določena z doložino lokalnosti. To je maksimalna razdalja med pticama ki še vedno vplivata druga na drugo. Za računanje novih stanj ptic v neki celici so tako potrebne le ptice v isti in sosednjih celicah. Uporabi se 8-kratna sosednost. Tako znatno pohitrimo izvajanje algoritma saj ni več potrebno iskati lokalnih ptic.

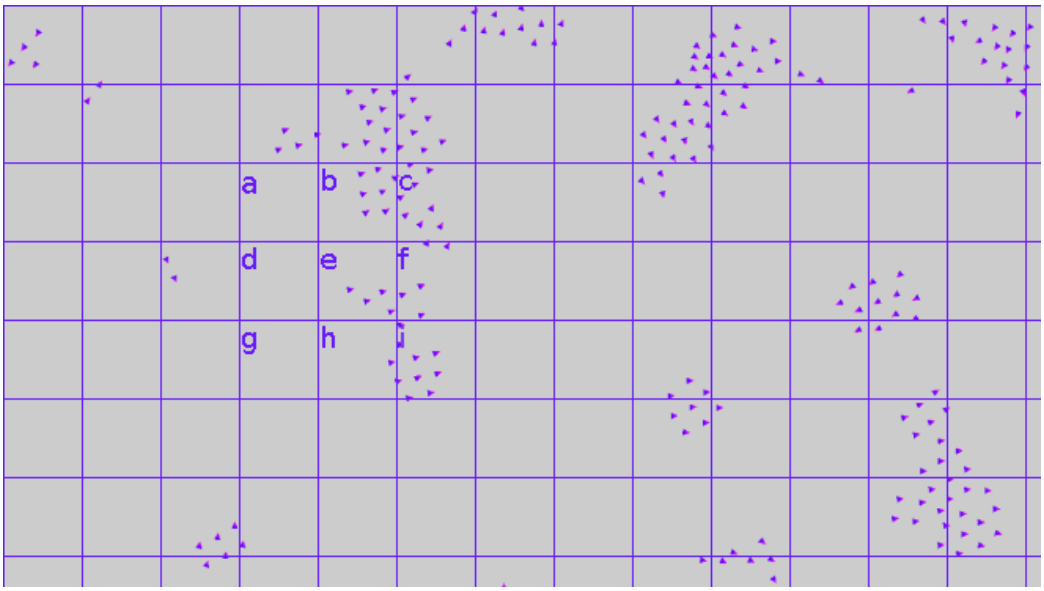
Rezultati meritev so tabelirani v Tabela 2.1. Algoritem z mrežo je opazno hitrejši. Pri 1000 pticah je algoritem z mrežo skoraj 6 krat hitrejši. Meritve so vizualizirane na Slika 2.3

Velikost jate	**Serijski algoritem [FPS]**	**Serijski algoritem z mrežo[FPS]**
100	653,222	1533,090
200	526,485	1347,730
300	390,755	1096,380
400	276,068	922,016
500	194,607	774,890
600	149,196	658,868
700	120,471	537,585
800	100,174	465,814
900	84.070	441,535
1000	71,061	419,116

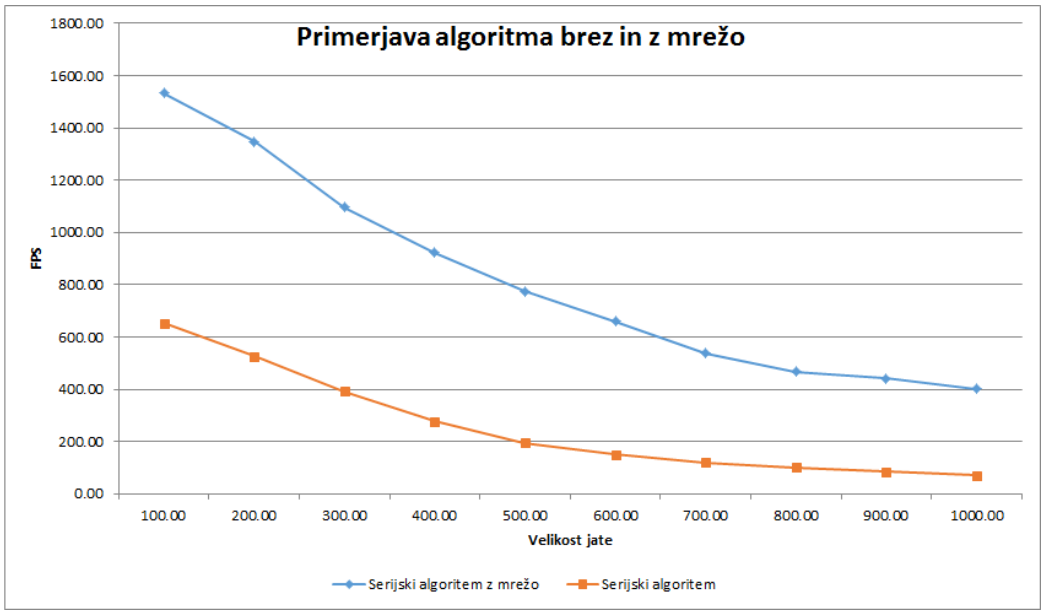
Tabela 2.1: Primerjava števila okvirjev na sekundo (ang. Frames per second) v odvisnosti od velikosti jate za serijski algoritem in serijski algoritem z mrežo

Algorithm 2 Groba psevdo koda serijskega algoritma z uporabo mreže

```
1: N ← Stevilo ptic
2: trenutnoStanje ← ptice[N]
3: naslendjeStanje ← ptice[N]
4: loop
5:     mrezaPtic ← razporediPticeVMrezo(trenutnoStanje);
6:     for i ← 0; i < N; i++ do
7:         naslendjeStanje[i] ← novoStanje(trenutnoStanje[i], mrezaPtic);
8:     izrisiStanje(naslendjeStanje);
9:     trenutnoStanje[i] ← naslendjeStanje;
```



Slika 2.2: Ilustracija mrežne. Za računanje nove pozicije se uporabi 8-kratna sosednjost. To pomeni, da se za izračun novi stanj ptic v celici e uporabijo ptice iz celic $a, b, c, d, e, f, g, h, i$



Slika 2.3: Graf primerja število okvirjev na sekundo za serijski algoritem z in brez mreže v odvisnosti od velikosti jate.

Poglavje 3

Paralelna implementacija z uporabo pThreads

TODO...

	Serijski	**pThreads - 2 niti**			**pThreads - 4 niti**			**pThreads - 8 niti**		
Velikost jate	FPS	FPS	Pohitritev	Učinkovitos	FPS	Pohitritev	Učinkovitos	FPS	Pohitritev	Učinkovitos
1000	419,12	401,96	0,96	0,48	437,63	1,04	0,26	409,84	0,98	0,12
1200	354,73	349,46	0,99	0,49	379,77	1,07	0,27	363,85	1,03	0,13
1400	296,02	311,61	1,05	0,53	312,54	1,06	0,26	331,14	1,12	0,14
1600	249,40	274,55	1,10	0,55	258,45	1,04	0,26	270,84	1,09	0,14
1800	216,03	242,16	1,12	0,56	232,21	1,07	0,27	225,42	1,04	0,13
2000	186,09	217,36	1,17	0,58	210,56	1,13	0,28	204,04	1,10	0,14
2200	163,80	195,88	1,20	0,60	191,45	1,17	0,29	184,85	1,13	0,14
2400	143,49	170,23	1,19	0,59	177,42	1,24	0,31	169,70	1,18	0,15
2600	129,57	146,25	1,13	0,56	160,77	1,24	0,31	155,48	1,20	0,15
2800	116,04	134,75	1,16	0,58	147,77	1,27	0,32	143,54	1,24	0,15

Tabela 3.1: My caption

Poglavje 4

Paralelna implementacija z uporabo OpenMP

TODO...

Poglavje 5

Paralelna implementacija z uporabo OpenCL

TODO...

Poglavje 6

Paralelna implementacija z uporabo MPI

TODO...

Literatura

[1] Michael Riis Andersen, Thomas Jensen, Pavel Lisouski, Anders Krogh Mortensen, Mikkel Kragh Hansen, Torben Gregersen, and Peter Ahrendt. Kinect depth sensor evaluation for computer vision applications. Technical report, Department of Engineering, Aarhus University, 2012.

[2] Andreja Balon. Vizualizacija. Diplomska naloga, Fakulteta za elektrotehniko in računalništvo, Univerza v Ljubljani, 1990.

[3] Lance Fortnow. Viewpoint time for computer science to grow up. *Communications of the ACM*, 52(8):33–35, 2009.

[4] Donald E Knuth and Peter B Bendix. Simple word problems in universal algebras. In Jörg H. Siekmann and Graham Wrightson, editors, *Automation of Reasoning: Classical papers on computational logic 1957–1966*, pages 342–376. Springer, 1983.

[5] Leslie Lamport. *LaTEX: A Document Preparation System*. Addison-Wesley, 1986.

[6] Oren Patashnik. BibTeXing. Dosegljivo: `http://bibtexml.sourceforge.net/btxdoc.pdf`, 1988. [Dostopano 5. 6. 2016].

[7] PDF/A. Dosegljivo: `http://en.wikipedia.org/wiki/PDF/A`, 2005. [Dostopano: 5. 6. 2016].

[8] Peter Peer and Borut Batagelj. Art—a perfect testbed for computer vision related research. In *Recent Advances in Multimedia Signal Processing and Communications*, pages 611–629. Springer, 2009.

[9] Franc Solina. 15 seconds of fame. *Leonardo*, 37(2):105–110, 2004.

[10] Franc Solina. Light fountain—an interactive art installation. Dosegljivo: `https://youtu.be/CS6x-QwJywg`, 2015. [Dostopano: 9. 10. 2015].