

Flussi Pedonali nell'Area del Colosseo: Codici

Roberto D'Autilia

2017-03-21

Abstract

File:colosseo_3_nuweb.tex

1 Introduzione

In questo lavoro vogliamo studiare i flussi pedonali nell'area del Colosseo nell'assetto urbanistico attuale e in quello di progetto. I luoghi dai quali si muove la popolazione sono l'uscita della metro M_1 , la Via Sacra M_2 e Via di San Gregorio M_3 . Abbiamo due carte, la situazione attuale (00-FASEO.jpg) e il progetto (00-FASE1.jpg).

La mappa 00-FASEO.jpg deve essere scalata per essere adattata alla simulazione. La scala ottimale è (0.27, 0.27).

Ora dobbiamo visualizzare le coordinate del mouse. Poi ricostruire tutte le aree chiuse. Le disegno in verde per non confonderle con quelle ancora non corrette.

Colosseo lo modellizziamo come due emicicli (poligoni chiusi in modo di avere un'entrata e un'uscita). Fatto questo cominciamo a simulare una popolazione di 1000 persone che entra nel Colosseo sulla prima entrata. Poiché la *drift* all'interno del Colosseo ci occorre anche un poligono area interna del Colosseo.

2 Codice

Vediamo come è fatto il codice. Le costanti sono

```
"../..../test/jl/costanti.jl" 1 ≡
```

```
# INIZIALIZZO LE COSTANTI
const N=1000 ::Int64          # Il numero di pedoni
const dt = 1.0 ::Float64      # Il passo di integrazione
const numero_iterazioni = 500 # Il numero di iterazioni della simulazione
const diag = sqrt(2) ::Float64 # diagonale
const passo = 1.0 ::Float64   # La dimensione di un passo
const raggio = 0.5 ::Float64  # Il raggio di non sovrapposizione dei pedoni
const dimenpedone = 2.0 ::Float64 # La dimensione del pedone
const scalax = 1.5 ::Float64  # lunghezza del passo di un pedone nella direzione x
const scalay = 1.5 ::Float64  # lunghezza del passo di un pedone nella direzione y
◇
```

```
"../test/jl/variabili_globali.jl" 2a ≡
```

```
# Variabili globali che sarebbe meglio eliminare
posizioni_prima = zeros(2,N)          # le posizioni dei pedoni al tempo t
#posizioni_dopo = rand(30.0:821.0,2,N) # le posizioni dei pedoni al tempo t+dt
altre = [rand(0.0:0.0,1,N);rand(100.0:100.0,1,N)] # altre posizioni
posizioni_dopo = hcat([rand(100.0:100.0,1,N);rand(200.0:200.0,1,N)],altre) # le posizioni dei pedoni al tempo
velocita = zeros(2,N)                  # le posizioni dei pedoni al tempo t+dt
k = 0 ::Int64                          # Un iteratore
passo = 1.0 ::Float64                  # La dimensione di un passo
raggio = 0.5 ::Float64                # Il raggio di non sovrapposizione dei pedoni
dimenpedone = 2.0 ::Float64           # La dimensione del pedone
scalax = 1.5 ::Float64                # lunghezza del passo di un pedone nella direzione x
scalay = 1.5 ::Float64                # lunghezza del passo di un pedone nella direzione y

mmm = 0.0 ::Float64                   # variabile di appoggio, ricontrollare se ci serve
◇
```

Per costruire i poligoni degli ostacoli dei pedoni costruiamo una costante di tipo dictionary

```
"../test/jl/edifici_coord.jl" 2b ≡
```

```
##### DICTIONARY POLIGONI DEGLI EDIFICI #####
#colosseo=[0 0 100 0; 100 0 100 200; 100 200 50 50; 50 50 0 100; 0 100 0 0]
##### DICTIONARY POLIGONI DEGLI EDIFICI #####
#colosseo=[0 0 100 0; 100 0 100 200; 100 200 50 50; 50 50 0 100; 0 100 0 0]
const edifici_coord = Dict{String, Array}(
"colosseo_coord" => [314 256; 329 242; 359 223; 382 212; 419 204; 454 200; 489 204; 523 211; 558 223; 597 245;
651 444; 631 482; 600 507; 562 525; 520 533; 485 531; 446 524; 412 510; 374 490; 344 462; 314 420; 300 384; 300 344; 270 312; 239 280; 208 248; 177 216; 146 184; 115 152; 84 120; 53 88; 22 56; 0 24; 0 52; 0 80; 0 108; 0 136; 0 164; 0 192; 0 220; 0 248; 0 276; 0 304; 0 332; 0 360; 0 388; 0 416; 0 444; 0 472; 0 500; 0 528; 0 556; 0 584; 0 612; 0 640; 0 668; 0 696; 0 724; 0 752; 0 780; 0 808; 0 836; 0 864; 0 892; 0 920; 0 948; 0 976; 0 1000], # L'emiciclo c
"colosseoa_coord" => [373 203;390 198; 410 192; 440 190; 464 190; 487 191; 507 194; 537 202; 567 213; 594 229; 621 245; 648 261; 675 277; 702 293; 729 309; 756 325; 783 341; 810 357; 837 373; 864 389; 891 405; 918 421; 945 437; 972 453; 999 469; 1026 485; 1053 501; 1080 517; 1107 533; 1134 549; 1161 565; 1188 581; 1215 597; 1242 613; 1269 629; 1296 645; 1323 661; 1350 677; 1377 693; 1404 709; 1431 725; 1458 741; 1485 757; 1512 773; 1539 789; 1566 805; 1593 821; 1620 837; 1647 853; 1674 869; 1701 885; 1728 901; 1755 917; 1782 933; 1809 949; 1836 965; 1863 981; 1890 997; 1917 1013; 1944 1029; 1971 1045; 1998 1061; 2025 1077; 2052 1093; 2079 1109; 2106 1125; 2133 1141; 2160 1157; 2187 1173; 2214 1189; 2241 1205; 2268 1221; 2295 1237; 2322 1253; 2349 1269; 2376 1285; 2403 1301; 2430 1317; 2457 1333; 2484 1349; 2511 1365; 2538 1381; 2565 1397; 2592 1413; 2619 1429; 2646 1445; 2673 1461; 2700 1477; 2727 1493; 2754 1509; 2781 1525; 2808 1541; 2835 1557; 2862 1573; 2889 1589; 2916 1605; 2943 1621; 2970 1637; 2997 1653; 3024 1669; 3051 1685; 3078 1701; 3105 1717; 3132 1733; 3159 1749; 3186 1765; 3213 1781; 3240 1797; 3267 1813; 3294 1829; 3321 1845; 3348 1861; 3375 1877; 3402 1893; 3429 1909; 3456 1925; 3483 1941; 3510 1957; 3537 1973; 3564 1989; 3591 2005; 3618 2021; 3645 2037; 3672 2053; 3699 2069; 3726 2085; 3753 2101; 3780 2117; 3807 2133; 3834 2149; 3861 2165; 3888 2181; 3915 2197; 3942 2213; 3969 2229; 3996 2245; 4023 2261; 4050 2277; 4077 2293; 4104 2309; 4131 2325; 4158 2341; 4185 2357; 4212 2373; 4239 2389; 4266 2405; 4293 2421; 4320 2437; 4347 2453; 4374 2469; 4401 2485; 4428 2501; 4455 2517; 4482 2533; 4509 2549; 4536 2565; 4563 2581; 4590 2597; 4617 2613; 4644 2629; 4671 2645; 4698 2661; 4725 2677; 4752 2693; 4779 2709; 4806 2725; 4833 2741; 4860 2757; 4887 2773; 4914 2789; 4941 2805; 4968 2821; 4995 2837; 5022 2853; 5049 2869; 5076 2885; 5103 2901; 5130 2917; 5157 2933; 5184 2949; 5211 2965; 5238 2981; 5265 2997; 5292 3013; 5319 3029; 5346 3045; 5373 3061; 5400 3077; 5427 3093; 5454 3109; 5481 3125; 5508 3141; 5535 3157; 5562 3173; 5589 3189; 5616 3205; 5643 3221; 5670 3237; 5697 3253; 5724 3269; 5751 3285; 5778 3301; 5805 3317; 5832 3333; 5859 3349; 5886 3365; 5913 3381; 5940 3397; 5967 3413; 5994 3429; 6021 3445; 6048 3461; 6075 3477; 6102 3493; 6129 3509; 6156 3525; 6183 3541; 6210 3557; 6237 3573; 6264 3589; 6291 3605; 6318 3621; 6345 3637; 6372 3653; 6399 3669; 6426 3685; 6453 3701; 6480 3717; 6507 3733; 6534 3749; 6561 3765; 6588 3781; 6615 3797; 6642 3813; 6669 3829; 6696 3845; 6723 3861; 6750 3877; 6777 3893; 6804 3909; 6831 3925; 6858 3941; 6885 3957; 6912 3973; 6939 3989; 6966 4005; 6993 4021; 7020 4037; 7047 4053; 7074 4069; 7101 4085; 7128 4101; 7155 4117; 7182 4133; 7209 4149; 7236 4165; 7263 4181; 7290 4197; 7317 4213; 7344 4229; 7371 4245; 7398 4261; 7425 4277; 7452 4293; 7479 4309; 7506 4325; 7533 4341; 7560 4357; 7587 4373; 7614 4389; 7641 4405; 7668 4421; 7695 4437; 7722 4453; 7749 4469; 7776 4485; 7803 4501; 7830 4517; 7857 4533; 7884 4549; 7911 4565; 7938 4581; 7965 4597; 7992 4613; 8019 4629; 8046 4645; 8073 4661; 8100 4677; 8127 4693; 8154 4709; 8181 4725; 8208 4741; 8235 4757; 8262 4773; 8289 4789; 8316 4805; 8343 4821; 8370 4837; 8397 4853; 8424 4869; 8451 4885; 8478 4901; 8505 4917; 8532 4933; 8559 4949; 8586 4965; 8613 4981; 8640 4997; 8667 5013; 8694 5029; 8721 5045; 8748 5061; 8775 5077; 8802 5093; 8829 5109; 8856 5125; 8883 5141; 8910 5157; 8937 5173; 8964 5189; 8991 5205; 9018 5221; 9045 5237; 9072 5253; 9099 5269; 9126 5285; 9153 5301; 9180 5317; 9207 5333; 9234 5349; 9261 5365; 9288 5381; 9315 5397; 9342 5413; 9369 5429; 9396 5445; 9423 5461; 9450 5477; 9477 5493; 9504 5509; 9531 5525; 9558 5541; 9585 5557; 9612 5573; 9639 5589; 9666 5605; 9693 5621; 9720 5637; 9747 5653; 9774 5669; 9801 5685; 9828 5701; 9855 5717; 9882 5733; 9909 5749; 9936 5765; 9963 5781; 9990 5797; 10017 5813; 10044 5829; 10071 5845; 10098 5861; 10125 5877; 10152 5893; 10179 5909; 10206 5925; 10233 5941; 10260 5957; 10287 5973; 10314 5989; 10341 6005; 10368 6021; 10395 6037; 10422 6053; 10449 6069; 10476 6085; 10503 6101; 10530 6117; 10557 6133; 10584 6149; 10611 6165; 10638 6181; 10665 6197; 10692 6213; 10719 6229; 10746 6245; 10773 6261; 10800 6277; 10827 6293; 10854 6309; 10881 6325; 10908 6341; 10935 6357; 10962 6373; 10989 6389; 11016 6405; 11043 6421; 11070 6437; 11097 6453; 11124 6469; 11151 6485; 11178 6501; 11205 6517; 11232 6533; 11259 6549; 11286 6565; 11313 6581; 11340 6597; 11367 6613; 11394 6629; 11421 6645; 11448 6661; 11475 6677; 11502 6693; 11529 6709; 11556 6725; 11583 6741; 11610 6757; 11637 6773; 11664 6789; 11691 6805; 11718 6821; 11745 6837; 11772 6853; 11799 6869; 11826 6885; 11853 6901; 11880 6917; 11907 6933; 11934 6949; 11961 6965; 11988 6981; 12015 6997; 12042 7013; 12069 7029; 12096 7045; 12123 7061; 12150 7077; 12177 7093; 12204 7109; 12231 7125; 12258 7141; 12285 7157; 12312 7173; 12339 7189; 12366 7205; 12393 7221; 12420 7237; 12447 7253; 12474 7269; 12501 7285; 12528 7301; 12555 7317; 12582 7333; 12609 7349; 12636 7365; 12663 7381; 12690 7397; 12717 7413; 12744 7429; 12771 7445; 12798 7461; 12825 7477; 12852 7493; 12879 7509; 12906 7525; 12933 7541; 12960 7557; 12987 7573; 13014 7589; 13041 7605; 13068 7621; 13095 7637; 13122 7653; 13149 7669; 13176 7685; 13203 7701; 13230 7717; 13257 7733; 13284 7749; 13311 7765; 13338 7781; 13365 7797; 13392 7813; 13419 7829; 13446 7845; 13473 7861; 13500 7877; 13527 7893; 13554 7909; 13581 7925; 13608 7941; 13635 7957; 13662 7973; 13689 7989; 13716 8005; 13743 8021; 13770 8037; 13797 8053; 13824 8069; 13851 8085; 13878 8101; 13905 8117; 13932 8133; 13959 8149; 13986 8165; 14013 8181; 14040 8197; 14067 8213; 14094 8229; 14121 8245; 14148 8261; 14175 8277; 14202 8293; 14229 8309; 14256 8325; 14283 8341; 14310 8357; 14337 8373; 14364 8389; 14391 8405; 14418 8421; 14445 8437; 14472 8453; 14499 8469; 14526 8485; 14553 8501; 14580 8517; 14607 8533; 14634 8549; 14661 8565; 14688 8581; 14715 8597; 14742 8613; 14769 8629; 14796 8645; 14823 8661; 14850 8677; 14877 8693; 14904 8709; 14931 8725; 14958 8741; 14985 8757; 15012 8773; 15039 8789; 15066 8805; 15093 8821; 15120 8837; 15147 8853; 15174 8869; 15201 8885; 15228 8901; 15255 8917; 15282 8933; 15309 8949; 15336 8965; 15363 8981; 15390 8997; 15417 9013; 15444 9029; 15471 9045; 15498 9061; 15525 9077; 15552 9093; 15579 9109; 15606 9125; 15633 9141; 15660 9157; 15687 9173; 15714 9189; 15741 9205; 15768 9221; 15795 9237; 15822 9253; 15849 9269; 15876 9285; 15903 9301; 15930 9317; 15957 9333; 15984 9349; 16011 9365; 16038 9381; 16065 9397; 16092 9413; 16119 9429; 16146 9445; 16173 9461; 16200 9477; 16227 9493; 16254 9509; 16281 9525; 16308 9541; 16335 9557; 16362 9573; 16389 9589; 16416 9605; 16443 9621; 16470 9637; 16497 9653; 16524 9669; 16551 9685; 16578 9701; 16605 9717; 16632 9733; 16659 9749; 16686 9765; 16713 9781; 16740 9797; 16767 9813; 16794 9829; 16821 9845; 16848 9861; 16875 9877; 16902 9893; 16929 9909; 16956 9925; 16983 9941; 17010 9957; 17037 9973; 17064 9989; 17091 10005; 17118 10021; 17145 10037; 17172 10053; 17199 10069; 17226 10085; 17253 10101; 17280 10117; 17307 10133; 17334 10149; 17361 10165; 17388 10181; 17415 10197; 17442 10213; 17469 10229; 17496 10245; 17523 10261; 17550 10277; 17577 10293; 17604 10309; 17631 10325; 17658 10341; 17685 10357; 17712 10373; 17739 10389; 17766 10405; 17793 10421; 17820 10437; 17847 10453; 17874 10469; 17901 10485; 17928 10501; 17955 10517; 17982 10533; 18009 10549; 18036 10565; 18063 10581; 18090 10597; 18117 10613; 18144 10629; 18171 10645; 18198 10661; 18225 10677; 18252 10693; 18279 10709; 18306 10725; 18333 10741; 18360 10757; 18387 10773; 18414 10789; 18441 10805; 18468 10821; 18495 10837; 18522 10853; 18549 10869; 18576 10885; 18603 10901; 18630 10917; 18657 10933; 18684 10949; 18711 10965; 18738 10981; 18765 11000; 18792 11016; 18819 11032; 18846 11048; 18873 11064; 18900 11080; 18927 11096; 18954 11112; 18981 11128; 19008 11144; 19035 11160; 19062 11176; 19089 11192; 19116 11208; 19143 11224; 19170 11240; 19197 11256; 19224 11272; 19251 11288; 19278 11304; 19305 11320; 19332 11336; 19359 11352; 19386 11368; 19413 11384; 19440 11400; 19467 11416; 19494 11432; 19521 11448; 19548 11464; 19575 11480; 19602 11496; 19629 11512; 19656 11528; 19683 11544; 19710 11560; 19737 11576; 19764 11592; 19791 11608; 19818 11624; 19845 11640; 19872 11656; 19899 11672; 19926 11688; 19953 11704; 19980 11720; 20007 11736; 20034 11752; 20061 11768; 20088 11784; 20115 11800; 20142 11816; 20169 11832; 20196 11848; 20223 11864; 20250 11880; 20277 11896; 20304 11912; 20331 11928; 20358 11944; 20385 11960; 20412 11976; 20439 11992; 20466 12008; 20493 12024; 20520 12040; 20547 12056; 20574 12072; 20601 12088; 20628 12104; 20655 12120; 20682 12136; 20709 12152; 20736 12168; 20763 12184; 20790 12200; 20817 12216; 20844 12232; 20871 12248; 20898 12264; 20925 12280; 20952 12296; 20979 12312; 21006 12328; 21033 12344; 21060 12360; 21087 12376; 21114 12392; 21141 12408; 21168 12424; 21195 12440; 21222 12456; 21249 12472; 21276 12488; 21303 12504; 21330 12520; 21357 12536; 21384 12552; 21411 12568; 21438 12584; 21465 12600; 21492 12616; 21519 12632; 21546 12648; 21573 12664; 21600 12680; 21627 12696; 21654 12712; 21681 12728; 21708 12744; 21735 12760; 21762 12776; 21789 12792; 21816 12808; 21843 12824; 21870 12840; 21897 12856; 21924 12872; 21951 12888; 21978 12904; 22005 12920; 22032 12936; 22059 12952; 22086 12968; 22113 12984; 22140 13000; 22167 13016; 22194 13032; 22221 13048; 22248 13064; 22275 13080; 22302 13096; 22329 13112; 22356 13128; 22383 13144; 22410 13160; 22437 13176; 22464 13192; 22491 13208; 22518 13224; 22545 13240; 22572 13256; 22599 13272; 22626 13288; 22653 13304; 22680 13320; 22707 13336; 22734 13352; 22761 13368; 22788 13384; 22815 13400; 22842 13416; 22869 13432; 22896 13448; 22923 13464; 22950 13480; 22977 13496; 23004 13512; 23031 13528; 23058 13544; 23085 13560; 23112 13576; 23139 13592; 23166 13608; 23193 13624; 23220 13640; 23247 13656; 23274 13672; 23301 13688; 23328 13704; 23355 13720; 23382 13736; 23409 13752; 23436 13768; 23463 13784; 23490 13800; 23517 13816; 23544 13832; 23571 13848; 23598 13864; 23625 13880; 23652 13896; 23679 13912; 23706 13928; 23733 13944; 23760 13960; 23787 13976; 23814 13992; 23841 14008; 23868 14024; 23895 14040; 23922 14056; 23949 14072; 23976 14088; 24003 14104; 24030 14120; 24057 14136; 24084 14152; 24111 14168; 24138 14184; 24165 14200; 24192 14216; 24219 14232; 24246 14248; 24273 14264; 24300 14280; 24327 14296; 24354 14312; 24381 14328; 24408 14344; 24435 14360; 24462 14376; 24489 14392; 24516 14408; 24543 14424; 24570 14440; 24597 14456; 24624 14472; 24651 14488; 24678 14504; 24705 14520; 24732 14536; 24759 14552; 24786 14568; 24813 14584; 24840 14600; 24867 14616; 24894 14632; 24921 14648; 24948 14664; 24975 14680; 25002 14696; 25029 14712; 25056 14728; 25083 14744; 25110 14760; 25137 14776; 25164 14792; 25191 14808; 25218 14824; 25245 14840; 25272 14856; 25299 14872; 25326 14888; 25353 14904; 25380 14920; 25407 14936; 25434 14952; 25461 14968; 25488 14984; 25515 15000; 25542 15016; 255
```

"../..test/jl/disegna_poligono.jl" 3a ≡

```
##### UNA FUNZIONE CHE CREA I POLIGONI DEGLI EDIFICI #####
function disegna_poligono(polig_coord)
    the_shape = ConvexShape()
    set_pointcount(the_shape, size(polig_coord)[1])
    for i = 1:size(polig_coord)[1]
        set_point(the_shape, i-1, Vector2f(polig_coord[i,1], polig_coord[i,2]))
    end
    set_position(the_shape, Vector2f(0.0, 0.0))
    set_fillcolor(the_shape, SFML.transparent)
    set_outlinecolor(the_shape, SFML.green)
    set_outline_thickness(the_shape, 2)
    return the_shape
end
#####
◇
```

Abbiamo poi una funzione che verifica che le coordinate del pedone non siano all'interno di alcun poligono chiuso

"../..test/jl/esterno.jl" 3b ≡

```
##### UNA FUNZIONE CHE VERIFICA CHE UN PUNTO NON SIA INTERNO AD ALCUN POLIGONO #####
function esterno(lax, lay, polig_coord)
    w = map(ciclo_poligono, values(polig_coord))
    a = sum(map(x->inpoly(lax,lay,x),w))
    if a>0
        return 1
    else
        return 0
    end
end
#####
◇
```

Dobbiamo ora definire lo stato dei pedoni. Fino a questo momento lo stato dei pedoni era rappresentato dalle sole coordinate (x, y) . Ora vogliamo invece definire un tipo (o una struttura) con le coordinate spaziali e le coordinate dell'obiettivo. Nell'esempio seguente introduciamo il tipo Pedone e costruiamo un array di pedoni inizializzati a PEDONE_DEFAULT

```
"../../../../test/jl/pedone_test.jl" 4a ≡
```

```
type Pedone
    lax ::Float64
    lay::Float64
    lavx::Float64
    lavy::Float64
    ladestx::Float64
    ladesty::Float64

end
const PEDONE_DEFAULT = Pedone(0.1,0.1,0.1,0.1,0.1,0.1)

Pedone() = Pedone(PEDONE_DEFAULT)
\#ESEMPIO DI UN Array di 100 pedoni

popolazione = Array{Pedone,100}
for i in 1:100
    popolazione[i] = Pedone()
end
◇
```

File defined by 4ab.

2.1 la funzione posizioni

Definiamo una funzione che prende lo stato della popolazione di N individui, che è un array di Statopedone e restituisce la matrice $2 \times N$ delle posizioni. Questa funzione sarà utile per calcolare la sovrapposizione delle posizioni con i metodi di NearestNeighbors

```
"../../../../test/jl/pedone_test.jl" 4b ≡
```

```
function posizioni(stato::Statopedone)
    lex = []
    ley = []
    for i=1:N
        push!(lex,stato[i].lax)
    end
    for i=1:N
        push!(ley,stato[i].lay)
    end
    return transpose([lex ley])
end
posizioni (generic function with 2 methods)
◇
```

File defined by 4ab.