

Profile Report

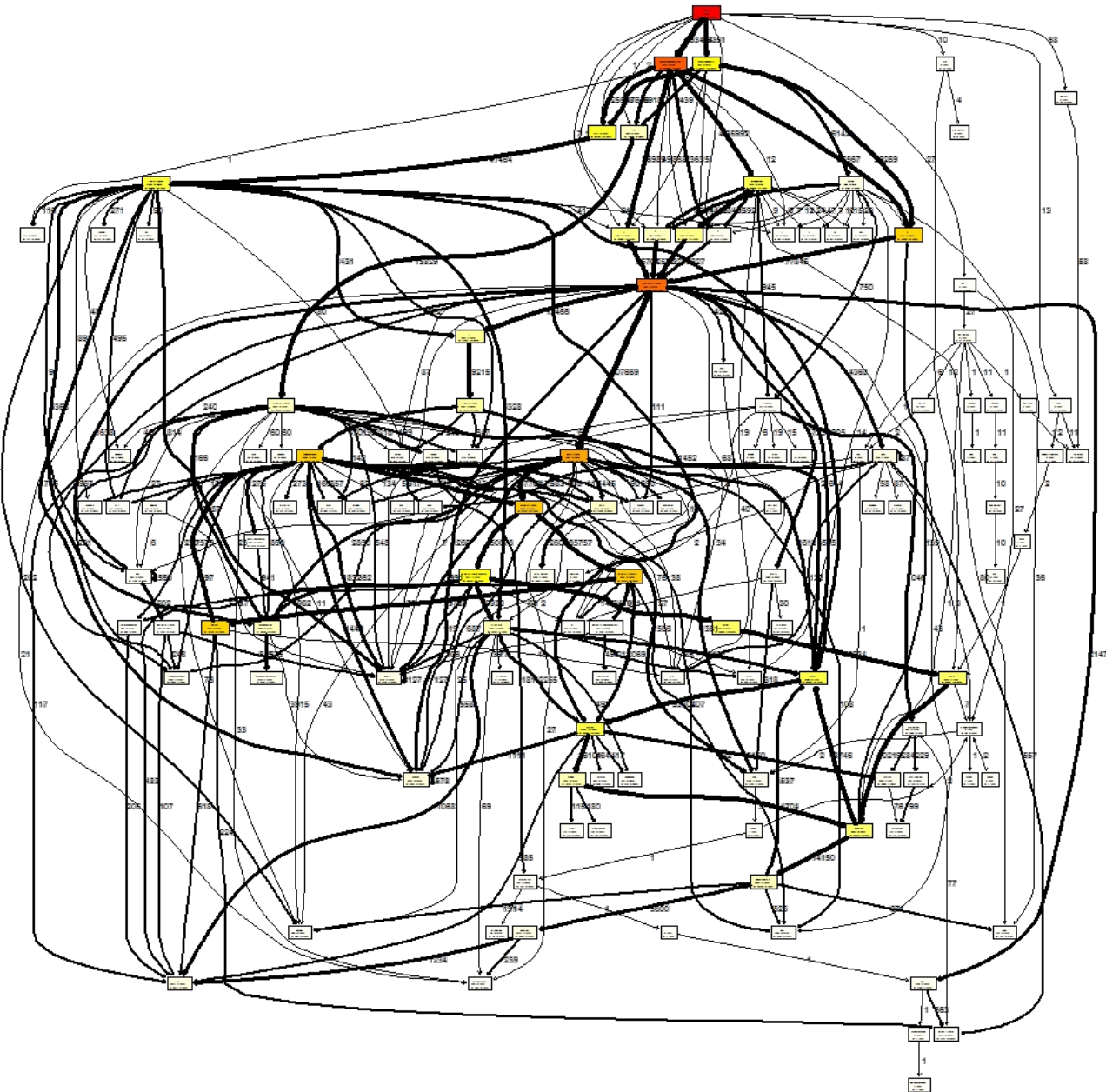
Summary

Table 1.

File	Total time	Selftime	Total time (%)	Selftime (%)
Simulation_NoChanges	9200	4200	220	100

Call graph

Figure 1.



Simulation_NoChanges

time	line
1	setwd("C:/Users/Johnny/Dropbox/Hood/DepthHonors/Rsim/RSimUsingSPH")
2	library("GUIProfiler")
3	
4	initPositions <- function(x, numParticles, numStars, starRadius, centers)

```

5   {
6   x = data.frame(star=rep(0, sum(numParticles)),
7   xPos=rep(0, sum(numParticles)),
8   yPos=rep(0, sum(numParticles)))
9
10  totalPartsPlaced = 1;
11
12  for(i in 1:numStars)
13  {
14    placedParticles = 1;
15
16    while(placedParticles <= numParticles[i])
17    {
18      tempX = runif(1, -(starRadius[i]), starRadius[i])
19      tempY = runif(1, -(starRadius[i]), starRadius[i])
20
21      tempPos = c(tempX, tempY)
22
23      if(sqrt(sum(tempPos^2)) <= starRadius[i])
24      {
25        x[totalPartsPlaced, ] <- c(i, tempPos[1] + centers[i, 1], tempPos[2] + centers[i, 2])
26        placedParticles = placedParticles + 1
27        totalPartsPlaced = totalPartsPlaced + 1
28      }
29    }
30  }
31  return(x)
32
33  }
34
35  initMasses <- function(m, numParticles, numStars, starMasses)
36  {
37    m = data.frame(star=rep(0, sum(numParticles)),
38    mass=rep(0, sum(numParticles)))
39
40    totalPartsPlaced = 1;
41
42    for(i in 1:numStars)
43    {
44      placedParticles = 1;
45
46      while(placedParticles <= numParticles[i])
47      {
48        partMass = starMasses[i]/numParticles[i]
49
50        m[totalPartsPlaced, ] <- c(i, partMass)
51        placedParticles = placedParticles + 1
52        totalPartsPlaced = totalPartsPlaced + 1
53      }
54    }
55    return(m)
56  }
57
58  initLambda <- function(lambda, numStars, starMass, starRadius, pressureConstant, PolyIndex)
59  {
60    lambda = c()
61
62    for(i in 1:numStars)
63    {
64      lambda <- c(lambda,
65      ((2*pressureConstant * (pi^(-1/PolyIndex))) *
66      ((starMass[i])*(1+PolyIndex)/((starRadius[i]^2))^(1+(1/PolyIndex))))/
67      starMass[i])
68    }
69
70    return(lambda)
71  }
72

```

```

73 #kernal functions
0.3 74 kernel <- function(position, smoothingLength, dimensions)
75 {
4.58 76 ch <- switch(
77 dimensions,
78 1/(6* smoothingLength),
79 5/(14 * pi * smoothingLength^2),
80 1/(4 * pi * smoothingLength^3)
81 )
82
91.98 83 q<-(sqrt(sum(position^2)))/smoothingLength
84
85 #Stops program if ch is null (ie not assigned by switch)
15.82 86 stopifnot(!is.null(ch))
87
1.16 88 if(is.na(q))
89 {
90 return(0)
91 }
92
3.52 93 if(q >= 0 && q < 1)
94 {
0.14 95 return (ch * ((2-q)^3 - 4*(1-q)^3))
96 }
97 else if(q >= 1 && q < 2)
98 {
0.14 99 return (ch * (2 -q)^3)
100 }
101 else if(q >= 2)
102 {
0.34 103 return(0)
104 }
105 }
106
0.52 107 gradKernel <- function(position, smoothingLength,dimensions)
108 {
4.8 109 ch <- switch(
110 dimensions,
111 1/(6* smoothingLength),
112 5/(14 * pi * smoothingLength^2),
113 1/(4 * pi * smoothingLength^3)
114 )
115
563.72 116 unitR <- position / (sqrt(sum(position^2)))
91.64 117 q<-(sqrt(sum(position^2)))/smoothingLength
118
119 #Stops program if ch is null (ie not assigned by switch)
19.86 120 stopifnot(!is.null(ch))
121
1.2 122 if(is.na(q))
123 {
124 return(0)
125 }
126
4.68 127 if(q >= 0 && q < 1)
128 {
4.24 129 return (ch * (1/smoothingLength) * (-3*(2-q)^2 + 12*(1-q)^2) * unitR)
130 }
131 else if(q >= 1 && q < 2)
132 {
23.74 133 return (ch * (1/smoothingLength) * (-3*(2 - q)^2) * unitR)
134 }
135 else if(q >= 2)
136 {
0.24 137 return(0)
138 }
139 }
140

```

```

141 #sudocode implementation
142 calculate_density <- function(x, m, h, rho, numParticles, dimensions)
143 {
144   for(i in 1:(numParticles-1)){
145     #initialize density with i = j contribution
1.66 146     rho[i, 1] <- m[i, 2] * kernel(0, h, dimensions)
147
0.52 148     for(j in (i+1):numParticles)
149     {
150       #calculate vector between two particles"
785.28 151       uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
157.98 152       rho_ij = m[i, 2] * kernel(uij, h, dimensions)
153
154       #add contribution to density"
95.3 155       rho[i, 1] <- rho[i, 1] + rho_ij
86.26 156       rho[j, 1] <- rho[j, 1] + rho_ij
157   }
158 }
159
160 return (rho)
161 }
162
163 calculate_Acceleration <- function(x, v, m, rho, P, nu, lambda, h, accel, numParticles, dimensions)
164 {
165   #add damping and gravity"
166   accel <- data.frame(xAccel=rep(0, numParticles),
167     yAccel=rep(0, numParticles))
168   if(dimensions == 3)
169   {
170     accel$zAccel <- rep(0, numParticles)
171   }
172
173   #not completly sure why it needs to be -1 but it ends up as a 101 row matrix and caues issues
174   for(i in 1:numParticles)
175   {
25.02 176     accel[i, 1:dimensions] <- -nu * v[i, 1:dimensions] - lambda[x[i, 1]] * x[i, 2:(dimensions+1)]
177   }
178
179   #add pressure"
180   for(i in 1:(numParticles-1))
181   {
1.12 182     for(j in (i+1):numParticles)
183     {
184       #calculate vector between two particles"
784.02 185       uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
186       #calculate acceleration due to pressure"
187
188       #Rprof("Profiling_GradKernel.txt", line.profiling = TRUE, append = TRUE)
868.66 189       p_a = (-m[j, 2])*(P[i, 1]/(rho[i, 1])^2 + P[j, 1]/(rho[j, 1])^2)*gradKernel(uij, h, dimensions)
190       #Rprof(NULL)
697.24 191       accel[i,] <- accel[i,] + p_a
692.02 192       accel[j,] <- accel[j,] - p_a
193
194   }
195 }
196
197 return(accel)
198 }
199
200 main <- function(){
201
202   #simulation Paramters
203   numParticles = c(120, 30)
204   totalParticles = sum(numParticles)
205
206   dimensions= 2
207   numStars = 2
208   starMass = c(1.6, .4)
209   starRadius = c(0.75,0.75)

```

```

209 smoothingLength = .04/sqrt(totalParticles/1000) #original .04/sqrt(numParticles/1000)
210 timeStep = .04
211 damping = 2.0
212 pressureConstant = 0.1
213 PolyIndex = 1
214 maxTimeSetps <- 250
215 profilingTimeSteps <- 100
216
217 centers = data.frame(x = c(0, 2),
218 y = c(0, 0))
219
220 rho = data.frame(rep(0, totalParticles))
221
222 #placeholders which will be set with init methods
223 x = 0
224 m = 0
225 lambda = 0
226
227 v = data.frame(xVel=rep(0, totalParticles),
228 yVel=rep(0, totalParticles))
229
230 accel = data.frame(xAccel=rep(0, totalParticles),
231 yAccel=rep(0, totalParticles))
232
233 #I think this needs to be calculated, but wasnt included in the code
234 #for now ill just assume that in our problem all particles are at rest for t < 0
235 v_mhalf = data.frame(xVel=rep(0, totalParticles),
236 yVel=rep(0, totalParticles))
237
238 v_phalf = data.frame(xVel=rep(0, totalParticles),
239 yVel=rep(0, totalParticles))
240
241 if(dimensions == 3)
242 {
243 x$zPos <- runif(totalParticles, -starRadius, starRadius)
244 v$zVel <- runif(totalParticles, -0.25, .25)
245 accel$zAccel <- rep(0, totalParticles)
246 v_mhalf$zVel <- zVel=runif(totalParticles, -0.25, .25)
247 v_phalf <- rep(0, totalParticles)
248 }
249
250
251 print(paste("Start ", Sys.time()))
252
253 x = initPositions(x, numParticles, numStars, starRadius, centers)
254 m = initMasses(m, numParticles, numStars, starMass)
255 lambda = initLambda(lambda, numStars, starMass, starRadius, pressureConstant , PolyIndex)
256
257
258 print(paste("Done initlizing particle positions at", Sys.time()))
259
260 png(file = './OutputPlots/_Start.png')
261 plot(x$ xPos, x$ yPos, xlim = c(-1, 3), ylim = c(-2, 2))
262 dev.off()
263
264 #Rprof("Profiling_GradKernel.txt", line.profiling = TRUE)
265 #print("Strating profiling")
266 #Rprof(NULL)
267
268 print("Starting main loop")
269
270 print("Strating profiling")
271
272 #RRprofStart(filename="GUIProfiling_GradKernel.txt")
273 #for(i in 1:maxTimeSetps)
274 #profilingOutput <- profvis({
0.02 275 # for(i in 1:profilingTimeSteps)
276 # {

```

```

0.02  277  # v_half = v_half + (accel * timeStep)
0.16  278  # x[c(2,3)] = x[c(2,3)] + v_half * timeStep
0.1    279  # v = .5 * (v_half + v_half)
      280  # v_half = v_half
      281
      282  # "update densities, pressures, accelerations"
      283  # print("Starting rho calc")
1127.02 284  # rho = calculate_density(x, m, smoothingLength, rho, totalParticles, dimensions)
      285  # print("Starting p calc")
0.02    286  # P = pressureConstant * rho^(1+1/PolyIndex)
      287  # ("Starting accel calc")
3068.08 288  # accel = calculate_Acceleration(x, v, m, rho, P, damping, lambda, smoothingLength, accel,
      289  # totalParticles, dimensions)
      290  # print("Starting save plot")
0.26    291  # png(file = paste("./OutputPlots/After", i, "loops.png", sep = ""))
0.54    292  # plot(x$ xPos, x$ yPos, xlim = c(-1, 3), ylim = c(-2, 2))
1.16    293  # dev.off()
      294
0.2      295  # print(paste("Done loop", i, "at", Sys.time(), sep=" "))
      296  # }
      297  #})
      298  #RRprofStop()
      299  print(paste("Done at", Sys.time()))
      300
      301  # print(profilingOutput)
      302  RRprofReport(file.name = "GUIProfiling_GradKernel.txt", reportname="GUIProfiling_GradKernel.html")
      303  #summaryRprof("Profiling_GradKernel.txt", lines="show")
      304
      305  }

```