Profile Report

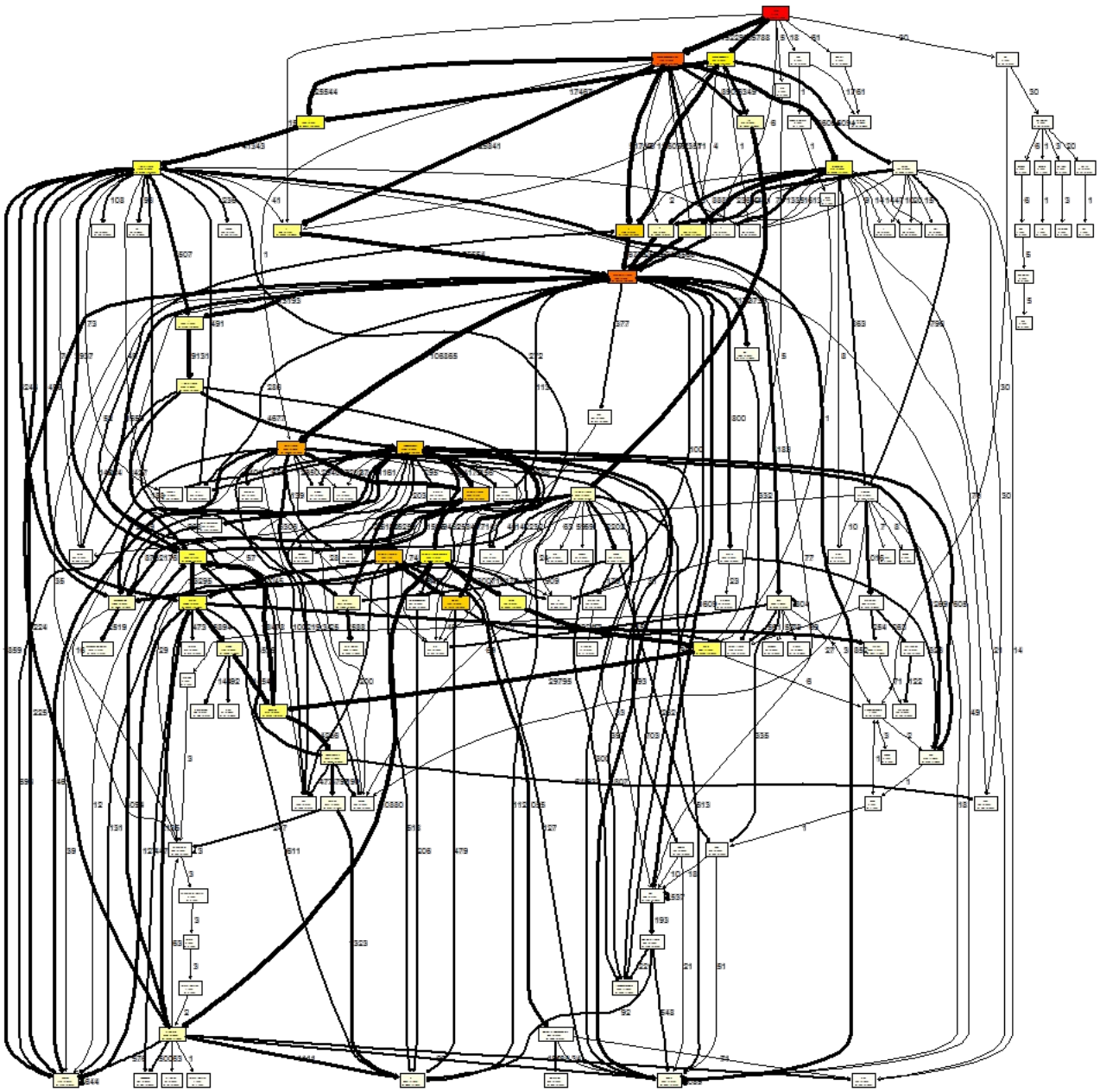
Summary

Table 1.

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

Call graph

Figure 1.



Simulation_NoChanges

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

```
{
x = data.frame(star=rep(0, sum(numParticles)),
xPos=rep(0, sum(numParticles)),
yPos=rep(0, sum(numParticles)))

totalPartsPlaced = 1;

for(i in 1:numStars)
{
placedParticles = 1;

while(placedParticles <= numParticles[i])
{
tempX = runif(1, -(starRadius[i]), starRadius[i])
tempY = runif(1, -(starRadius[i]), starRadius[i])

tempPos = c(tempX, tempY)

if(sqrt(sum(tempPos^2)) <= starRadius[i])
{
x[totalPartsPlaced, ] <- c(i, tempPos[1] + centers[i, 1], tempPos[2] + centers[i, 2])
placedParticles = placedParticles + 1
totalPartsPlaced = totalPartsPlaced + 1
}
}
}
return(x)

}

initMasses <- function(m, numParticles, numStars, starMasses)
{
m = data.frame(star=rep(0, sum(numParticles)),
mass=rep(0, sum(numParticles)))

totalPartsPlaced = 1;

for(i in 1:numStars)
{
placedParticles = 1;

while(placedParticles <= numParticles[i])
{
partMass = starMasses[i]/numParticles[i]

m[totalPartsPlaced, ] <- c(i, partMass)
placedParticles = placedParticles + 1
totalPartsPlaced = totalPartsPlaced + 1
}
}
return(m)
}

initLambda <- function(lambda, numStars, starMass, starRadius, presureConstant, PolyIndex)
{
lambda = c()

for(i in 1:numStars)
{
lambda <- c(lambda,
((2*presureConstant * (pi^(-1/PolyIndex))) *
((starMass[i])*(1+PolyIndex)/((starRadius[i])^2))^(1+(1/PolyIndex)))/
starMass[i])
}

return(lambda)
}

```

```
        73  #kernal functions
0.5     74  kernel <- function(position, smoothingLength, dimensions)
        75  {
3.76    76  ch <- switch(
        77  dimensions,
        78  1/(6* smoothingLength),
        79  5/(14 * pi * smoothingLength^2),
        80  1/(4 * pi * smoothingLength^3)
        81  )
        82
91.12   83  q<-(sqrt(sum(position^2)))/smoothingLength
        84
        85  #Stops program if ch is null (ie not assigned by switch)
16.26   86  stopifnot(!is.null(ch))
        87
1.22    88  if(is.na(q))
        89  {
        90  return(0)
        91  }
        92
3.44    93  if(q >= 0 && q < 1)
        94  {
0.06    95  return (ch *((2-q)^3 - 4*(1-q)^3))
        96  }
        97  else if(q >= 1 && q < 2)
        98  {
0.06    99  return (ch * (2 -q)^3)
       100  }
       101  else if(q >= 2)
       102  {
0.28   103  return(0)
       104  }
       105  }
       106
0.44   107  gradKernel <- function(position, smoothingLength,dimensions)
       108  {
4.84   109  ch <- switch(
       110  dimensions,
       111  1/(6* smoothingLength),
       112  5/(14 * pi * smoothingLength^2),
       113  1/(4 * pi * smoothingLength^3)
       114  )
       115
566.52 116  unitR <- position / (sqrt(sum(position^2)))
91.34  117  q<-(sqrt(sum(position^2)))/smoothingLength
       118
       119  #Stops program if ch is null (ie not assigned by switch)
0.94   120  if(is.null(ch))
       121  {
       122  print("null")
       123  }
18.04  124  stopifnot(!is.null(ch)) #need to look up a better way to stop execution
       125
1.3    126  if(is.na(q))
       127  {
       128  return(0)
       129  }
       130
3.98   131  if(q >= 0 && q < 1)
       132  {
4.26   133  return (ch * (1/smoothingLength) * (-3*(2-q)^2 + 12*(1-q)^2) * unitR)
       134  }
       135  else if(q >= 1 && q < 2)
       136  {
22.68  137  return (ch * (1/smoothingLength) * (-3*(2 - q)^2) * unitR)
       138  }
       139  else if(q >= 2)
       140  {
```

```
0.36   141  return(0)
       142  }
       143  }
       144
       145  #sudocode implemtation
       146  calculate_density <- function(x, m, h, rho, numParticles, dimensions)
       147  {
       148  for(i in 1:(numParticles-1)){
       149  #"initialize density with i = j contribution"
1.72   150  rho[i, 1] <- m[i, 2] * kernel(0, h, dimensions)
       151
0.66   152  for(j in (i+1):numParticles)
       153  {
       154  #"calculate vector between two particles"
778.52 155  uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
157.22 156  rho_ij = m[i, 2] * kernel(uij, h, dimensions)
       157
       158  #"add contribution to density"
92.96  159  rho[i, 1] <- rho[i, 1] + rho_ij
84.66  160  rho[j, 1] <- rho[j, 1] + rho_ij
       161  }
       162  }
       163
       164  return (rho)
       165  }
       166
       167  calculate_Acceleration <- function(x, v, m, rho, P, nu, lambda, h, accel, numParticles, dimensions)
       168  {
       169  #RRprofStart(filename="GUIProfiling_Accel.txt")
       170  #"add damping and gravity"
0.02   171  accel <- data.frame(xAccel=rep(0, numParticles),
       172  yAccel=rep(0, numParticles))
       173  if(dimensions == 3)
       174  {
       175  accel$zAccel <- rep(0, numParticles)
       176  }
       177
       178  #not completly sure why it needs to be -1 but it ends up as a 101 row matrix and caues issues
       179  for(i in 1:numParticles)
       180  {
24.62  181  accel[i, 1:dimensions] <- -nu * v[i, 1:dimensions] - lambda[x[i, 1]] * x[i, 2:(dimensions+1)]
       182  }
       183
       184  #"add pressure"
       185  for(i in 1:(numParticles-1))
       186  {
0.92   187  for(j in (i+1):numParticles)
       188  {
       189  #"calculate vector between two particles"
778.02 190  uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
       191  #"calculate acceleration due to pressure"
       192
       193  #Rprof("Profiling_GradKernel.txt", line.profiling = TRUE, append = TRUE)
       194  #RRprofStart(filename="GUIProfiling_Gradkernel.txt")
868.32 195  p_a = (-m[j, 2])*(P[i, 1]/(rho[i, 1])^2 + P[j, 1]/(rho[j, 1])^2)*gradKernel(uij, h, dimensions)
       196
       197  #Rprof(NULL)
690.18 198  accel[i,] <- accel[i,] + p_a
683.64 199  accel[j,] <- accel[j,] - p_a
       200  #RRprofStop()
       201
       202  }
       203  }
       204
       205  #RRprofStop()
       206
       207  return(accel)
       208  }
```

```
209
210 main <- function(){
211
212 #simulation Paramters
213 numParticles = c(120, 30)
214 totalParticles = sum(numParticles)
215 dimensions= 2
216 numStars = 2
217 starMass = c(1.6, .4)
218 starRadius = c(0.75,0.75)
219 smoothingLength = .04/sqrt(totalParticles/1000) #orginal .04/sqrt(numParticles/1000)
220 timeStep = .04
221 damping = 2.0
222 presureConstant = 0.1
223 PolyIndex = 1
224 maxTimeSetps <- 250
225 profilingTimeSteps <- 100
226
227 centers = data.frame(x = c(0, 2),
228 y = c(0, 0))
229
230 rho = data.frame(rep(0, totalParticles))
231
232 #placeholders which will be set with init methods
233 x = 0
234 m = 0
235 lambda = 0
236
237 v = data.frame(xVel=rep(0, totalParticles),
238 yVel=rep(0, totalParticles))
239
240 accel = data.frame(xAccel=rep(0, totalParticles),
241 yAccel=rep(0, totalParticles))
242
243 #I think this needs to be calcuated, but wasnt included in the code
244 #for now ill just assume that in our problem all particles are at rest for t < 0
245 v_mhalf = data.frame(xVel=rep(0, totalParticles),
246 yVel=rep(0, totalParticles))
247
248 v_phalf = data.frame(xVel=rep(0, totalParticles),
249 yVel=rep(0, totalParticles))
250
251 if(dimensions == 3)
252 {
253 x$zPos <- runif(totalParticles, -starRadius, starRadius)
254 v$zVel <- runif(totalParticles, -0.25, .25)
255 accel$zAccel <- rep(0, totalParticles)
256 v_mhalf$zVel <- zVel=runif(totalParticles, -0.25, .25)
257 v_phalf <- rep(0, totalParticles)
258 }
259
260
261 print(paste("Start ",Sys.time()))
262
263 x = initPositions(x, numParticles,numStars, starRadius, centers)
264 m = initMasses(m, numParticles, numStars, starMass)
265 lambda = initLambda(lambda, numStars, starMass, starRadius, presureConstant , PolyIndex)
266
267
268 print(paste("Done initlizing particle positions at", Sys.time()))
269
270 png(file = './OutputPlots/_Start.png')
271 plot(x$xPos, x$yPos, xlim = c(-1, 3), ylim = c(-2, 2))
272 dev.off()
273
274 #Rprof("Profiling_GradKernel.txt", line.profiling = TRUE)
275 #print("Strating profiling")
276 #Rprof(NULL)
```

```
277
278 print("Starting main loop")
279
280 print("Strating profiling")
281
282 RRprofStart(filename="GUIProfiling_FullGradKernel.txt")
283 #for(i in 1:maxTimeSetps)
284 for(i in 1:profilingTimeSteps)
285 {
0.16 286 v_phalf = v_mhalf + (accel * timeStep)
0.12 287 x[c(2,3)] = x[c(2,3)] + v_phalf * timeStep
0.14 288 v = .5 * (v_mhalf + v_phalf)
289 v_mhalf = v_phalf
290
291 #"update densities, pressures, accelerations"
1115.76 292 rho = calculate_density(x, m, smoothingLength, rho, totalParticles, dimensions)
293 P = presureConstant * rho^(1+1/PolyIndex)
3045.8 294 accel = calculate_Acceleration(x, v, m, rho, P, damping, lambda, smoothingLength, accel,
totalParticles, dimensions)
295
0.36 296 png(file = paste("./OutputPlots/After", i,"loops.png", sep = ""))
0.6 297 plot(x$xPos, x$yPos, xlim = c(-1, 3), ylim = c(-2, 2))
1.22 298 dev.off()
299
0.1 300 print(paste("Done loop", i, "at",Sys.time(), sep=" "))
301 }
302 RRprofStop()
303
304 print(paste("Done at", Sys.time()))
305 RRprofReport(file.name = "GUIProfiling_FullGradKernel.txt",
reportname="GUIProfiling_FullGradKernel.html")
306 }
```