Profile Report

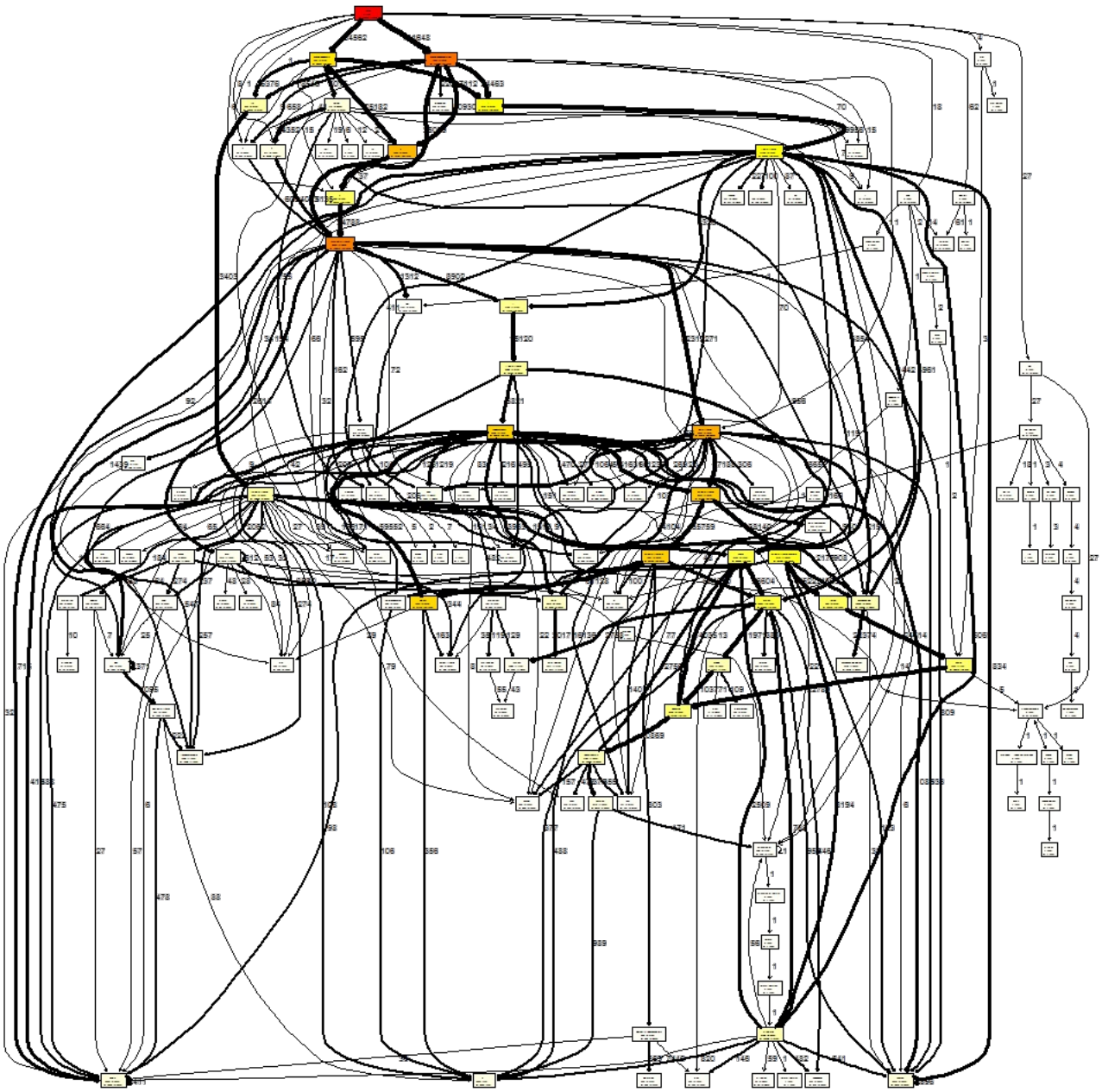
Summary

Table 1.

File	Total time	Self time	Total time (%)	Self time (%)
Simulation_NoChanges	6800	3300	200	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
0.3   74  kernel <- function(position, smoothingLength, dimensions)
      75  {
4.66  76  ch <- switch(
      77  dimensions,
      78  1/(6* smoothingLength),
      79  5/(14 * pi * smoothingLength^2),
      80  1/(4 * pi * smoothingLength^3)
      81  )
      82
89.68 83  q<-(sqrt(sum(position^2)))/smoothingLength
      84
      85  #Stops program if ch is null (ie not assigned by switch)
15.74 86  stopifnot(!is.null(ch))
      87
0.94  88  if(is.na(q))
      89  {
      90  return(0)
      91  }
      92
2.92  93  if(q >= 0 && q < 1)
      94  {
0.4   95  return (ch *((2-q)^3 - 4*(1-q)^3))
      96  }
      97  else if(q >= 1 && q < 2)
      98  {
0.44  99  return (ch * (2 -q)^3)
     100  }
     101  else if(q >= 2)
     102  {
0.26 103  return(0)
     104  }
     105  }
     106
0.32 107  gradKernel <- function(position, smoothingLength,dimensions)
     108  {
0.5  109  return(1)
     110  # ch <- switch(
     111  # dimensions,
     112  # 1/(6* smoothingLength),
     113  # 5/(14 * pi * smoothingLength^2),
     114  # 1/(4 * pi * smoothingLength^3)
     115  # )
     116
     117  # unitR <- position / (sqrt(sum(position^2)))
     118  # q<-(sqrt(sum(position^2)))/smoothingLength
     119
     120  #Stops program if ch is null (ie not assigned by switch)
     121  # if(is.null(ch))
     122  # {
     123  # print("null")
     124  # }
     125  # stopifnot(!is.null(ch)) #need to look up a better way to stop execution
     126
     127  # if(is.na(q))
     128  # {
     129  # return(0)
     130  # }
     131
     132  # if(q >= 0 && q < 1)
     133  # {
     134  # return (ch * (1/smoothingLength) * (-3*(2-q)^2 + 12*(1-q)^2) * unitR)
     135  # }
     136  # else if(q >= 1 && q < 2)
     137  # {
     138  # return (ch * (1/smoothingLength) * (-3*(2 - q)^2) * unitR)
     139  # }
     140  # else if(q >= 2)
```

```
        141 # {
        142 # return(0)
        143 # }
        144 }
        145
        146 #sudocode implemtation
        147 calculate_density <- function(x, m, h, rho, numParticles, dimensions)
        148 {
        149 for(i in 1:(numParticles-1)){
        150 #"initialize density with i = j contribution"
1.66    151 rho[i, 1] <- m[i, 2] * kernel(0, h, dimensions)
        152
0.42    153 for(j in (i+1):numParticles)
        154 {
        155 #"calculate vector between two particles"
752.9   156 uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
154.84  157 rho_ij = m[i, 2] * kernel(uij, h, dimensions)
        158
        159 #"add contribution to density"
94.06   160 rho[i, 1] <- rho[i, 1] + rho_ij
87.32   161 rho[j, 1] <- rho[j, 1] + rho_ij
        162 }
        163 }
        164
        165 return (rho)
        166 }
        167
        168 calculate_Acceleration <- function(x, v, m, rho, P, nu, lambda, h, accel, numParticles, dimensions)
        169 {
        170 #RRprofStart(filename="GUIProfiling_Accel.txt")
        171 #"add damping and gravity"
0.02    172 accel <- data.frame(xAccel=rep(0, numParticles),
        173 yAccel=rep(0, numParticles))
        174 if(dimensions == 3)
        175 {
        176 accel$zAccel <- rep(0, numParticles)
        177 }
        178
        179 #not completly sure why it needs to be -1 but it ends up as a 101 row matrix and caues issues
        180 for(i in 1:numParticles)
        181 {
24.12   182 accel[i, 1:dimensions] <- -nu * v[i, 1:dimensions] - lambda[x[i, 1]] * x[i, 2:(dimensions+1)]
        183 }
        184
        185 #"add pressure"
0.02    186 for(i in 1:(numParticles-1))
        187 {
0.78    188 for(j in (i+1):numParticles)
        189 {
        190 #"calculate vector between two particles"
759.06  191 uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
        192 #"calculate acceleration due to pressure"
        193
        194 #Rprof("Profiling_GradKernel.txt", line.profiling = TRUE, append = TRUE)
        195 #RRprofStart(filename="GUIProfiling_Gradkernel.txt")
113.48  196 p_a = (-m[j, 2])*(P[i, 1]/(rho[i, 1])^2 + P[j, 1]/(rho[j, 1])^2)*gradKernel(uij, h, dimensions)
        197
        198 #Rprof(NULL)
665.6   199 accel[i,] <- accel[i,] + p_a
669.84  200 accel[j,] <- accel[j,] - p_a
        201 #RRprofStop()
        202
        203 }
        204 }
        205
        206 #RRprofStop()
        207
        208 return(accel)
```

```
}

main <- function(){

#simulation Paramters
numParticles = c(120, 30)
totalParticles = sum(numParticles)
dimensions= 2
numStars = 2
starMass = c(1.6, .4)
starRadius = c(0.75,0.75)
smoothingLength = .04/sqrt(totalParticles/1000) #orginal .04/sqrt(numParticles/1000)
timeStep = .04
damping = 2.0
presureConstant = 0.1
PolyIndex = 1
maxTimeSetps <- 250
profilingTimeSteps <- 100

centers = data.frame(x = c(0, 2),
y = c(0, 0))

rho = data.frame(rep(0, totalParticles))

#placeholders which will be set with init methods
x = 0
m = 0
lambda = 0

v = data.frame(xVel=rep(0, totalParticles),
yVel=rep(0, totalParticles))

accel = data.frame(xAccel=rep(0, totalParticles),
yAccel=rep(0, totalParticles))

#I think this needs to be calcuated, but wasnt included in the code
#for now ill just assume that in our problem all particles are at rest for t < 0
v_mhalf = data.frame(xVel=rep(0, totalParticles),
yVel=rep(0, totalParticles))

v_phalf = data.frame(xVel=rep(0, totalParticles),
yVel=rep(0, totalParticles))

if(dimensions == 3)
{
x$zPos <- runif(totalParticles, -starRadius, starRadius)
v$zVel <- runif(totalParticles, -0.25, .25)
accel$zAccel <- rep(0, totalParticles)
v_mhalf$zVel <- zVel=runif(totalParticles, -0.25, .25)
v_phalf <- rep(0, totalParticles)
}


print(paste("Start ",Sys.time()))

x = initPositions(x, numParticles,numStars, starRadius, centers)
m = initMasses(m, numParticles, numStars, starMass)
lambda = initLambda(lambda, numStars, starMass, starRadius, presureConstant , PolyIndex)


print(paste("Done initlizing particle positions at", Sys.time()))

png(file = './OutputPlots/_Start.png')
plot(x$xPos, x$yPos, xlim = c(-1, 3), ylim = c(-2, 2))
dev.off()

#Rprof("Profiling_GradKernel.txt", line.profiling = TRUE)
#print("Strating profiling")
```

```
        277 #Rprof(NULL)
        278
        279 print("Starting main loop")
        280
        281 print("Strating profiling")
        282
        283 RRprofStart(filename="GUIProfiling_SimpleGradKernel.txt")
        284 #for(i in 1:maxTimeSetps)
        285 for(i in 1:profilingTimeSteps)
        286 {
0.12    287 v_phalf = v_mhalf + (accel * timeStep)
0.12    288 x[c(2,3)] = x[c(2,3)] + v_phalf * timeStep
0.06    289 v = .5 * (v_mhalf + v_phalf)
        290 v_mhalf = v_phalf
        291
        292 #"update densities, pressures, accelerations"
1091.24 293 rho = calculate_density(x, m, smoothingLength, rho, totalParticles, dimensions)
0.02    294 P = presureConstant * rho^(1+1/PolyIndex)
2232.96 295 accel = calculate_Acceleration(x, v, m, rho, P, damping, lambda, smoothingLength, accel,
            totalParticles, dimensions)
        296
0.36    297 png(file = paste("./OutputPlots/After", i,"loops.png", sep = ""))
0.54    298 plot(x$xPos, x$yPos, xlim = c(-1, 3), ylim = c(-2, 2))
1.24    299 dev.off()
        300
0.08    301 print(paste("Done loop", i, "at",Sys.time(), sep=" "))
        302 }
        303 RRprofStop()
        304
        305 print(paste("Done at", Sys.time()))
        306 RRprofReport(file.name = "GUIProfiling_SimpleGradKernel.txt",
            reportname="GUIProfiling_SimpleGradKernel.html")
        307 }
```