Profile Report

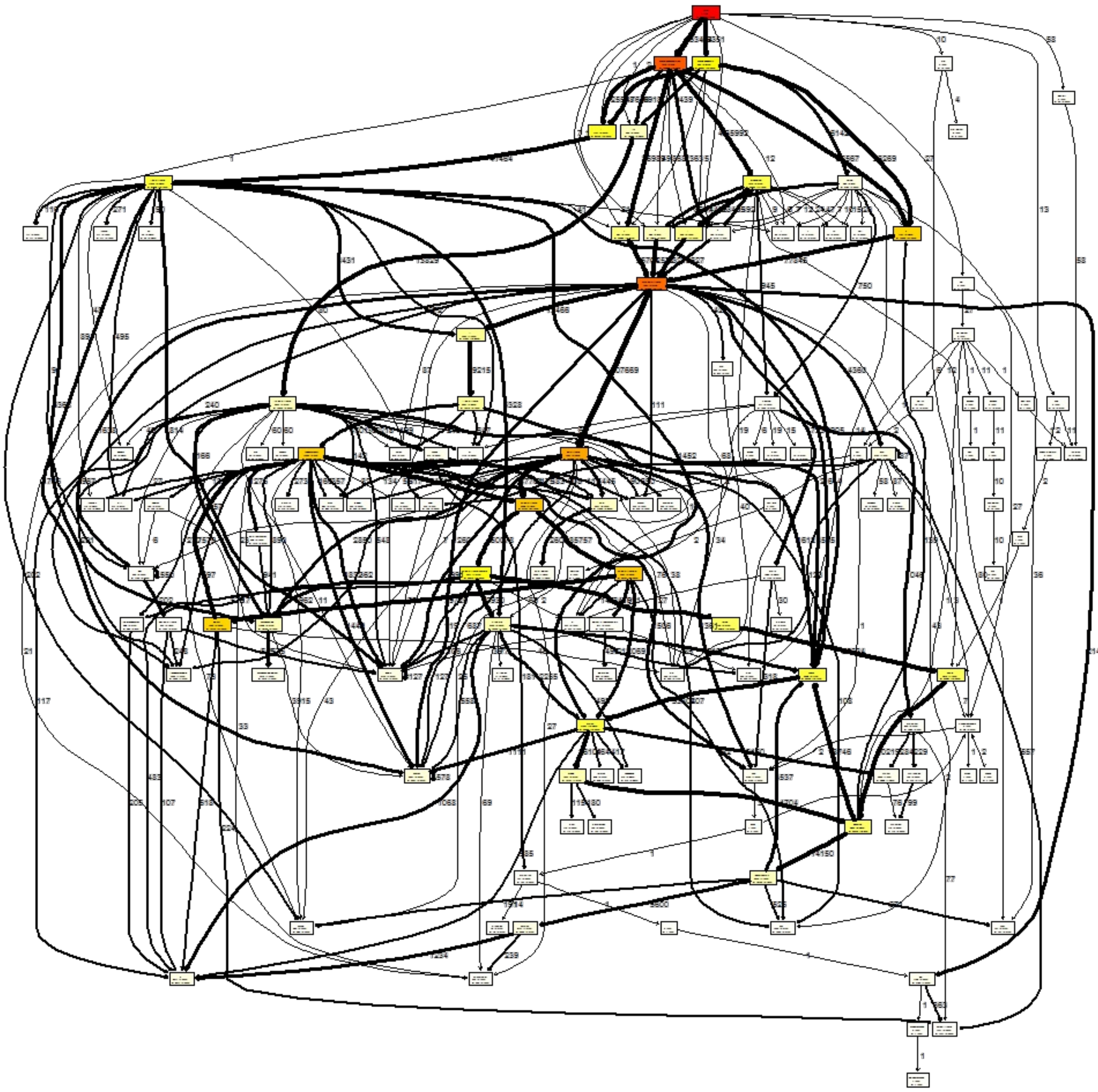
Summary

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)
}
```

	#kernal functions
0.3	kernel <- function(position, smoothingLength, dimensions)
	{
4.58	ch <- switch(
	dimensions,
	1/(6* smoothingLength),
	5/(14 * pi * smoothingLength^2),
	1/(4 * pi * smoothingLength^3)
	)
91.98	q<-(sqrt(sum(position^2)))/smoothingLength
	#Stops program if ch is null (ie not assigned by switch)
15.82	stopifnot(!is.null(ch))
1.16	if(is.na(q))
	{
	return(0)
	}
3.52	if(q >= 0 && q < 1)
	{
0.14	return (ch *((2-q)^3 - 4*(1-q)^3))
	}
	else if(q >= 1 && q < 2)
	{
0.14	return (ch * (2 -q)^3)
	}
	else if(q >= 2)
	{
0.34	return(0)
	}
	}
0.52	gradKernel <- function(position, smoothingLength,dimensions)
	{
4.8	ch <- switch(
	dimensions,
	1/(6* smoothingLength),
	5/(14 * pi * smoothingLength^2),
	1/(4 * pi * smoothingLength^3)
	)
563.72	unitR <- position / (sqrt(sum(position^2)))
91.64	q<-(sqrt(sum(position^2)))/smoothingLength
	#Stops program if ch is null (ie not assigned by switch)
19.86	stopifnot(!is.null(ch))
1.2	if(is.na(q))
	{
	return(0)
	}
4.68	if(q >= 0 && q < 1)
	{
4.24	return (ch * (1/smoothingLength) * (-3*(2-q)^2 + 12*(1-q)^2) * unitR)
	}
	else if(q >= 1 && q < 2)
	{
23.74	return (ch * (1/smoothingLength) * (-3*(2 - q)^2) * unitR)
	}
	else if(q >= 2)
	{
0.24	return(0)
	}
	}

```
#sudocode implemtation
calculate_density <- function(x, m, h, rho, numParticles, dimensions)
{
for(i in 1:(numParticles-1)){
#"initialize density with i = j contribution"
1.66    rho[i, 1] <- m[i, 2] * kernel(0, h, dimensions)

0.52    for(j in (i+1):numParticles)
{
#"calculate vector between two particles"
785.28  uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
157.98  rho_ij = m[i, 2] * kernel(uij, h, dimensions)

#"add contribution to density"
95.3    rho[i, 1] <- rho[i, 1] + rho_ij
86.26   rho[j, 1] <- rho[j, 1] + rho_ij
}
}

return (rho)
}

calculate_Acceleration <- function(x, v, m, rho, P, nu, lambda, h, accel, numParticles, dimensions)
{
#"add damping and gravity"
accel <- data.frame(xAccel=rep(0, numParticles),
yAccel=rep(0, numParticles))
if(dimensions == 3)
{
accel$zAccel <- rep(0, numParticles)
}

#not completly sure why it needs to be -1 but it ends up as a 101 row matrix and caues issues
for(i in 1:numParticles)
{
25.02   accel[i, 1:dimensions] <- -nu * v[i, 1:dimensions] - lambda[x[i, 1]] * x[i, 2:(dimensions+1)]
}

#"add pressure"
for(i in 1:(numParticles-1))
{
1.12    for(j in (i+1):numParticles)
{
#"calculate vector between two particles"
784.02  uij = x[i,2:(dimensions+1)] - x[j,2:(dimensions+1)]
#"calculate acceleration due to pressure"

#Rprof("Profiling_GradKernel.txt", line.profiling = TRUE, append = TRUE)
868.66  p_a = (-m[j, 2])*(P[i, 1]/(rho[i, 1])^2 + P[j, 1]/(rho[j, 1])^2)*gradKernel(uij, h, dimensions)
#Rprof(NULL)
697.24  accel[i,] <- accel[i,] + p_a
692.02  accel[j,] <- accel[j,] - p_a

}
}

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")
#Rprof(NULL)

print("Starting main loop")

print("Strating profiling")

#RRprofStart(filename="GUIProfiling_GradKernel.txt")
#for(i in 1:maxTimeSetps)
#profilingOutput <- profvis({
0.02  # for(i in 1:profilingTimeSteps)
# {
```

0.02	# v_phalf = v_mhalf + (accel * timeStep)
0.16	# x[c(2,3)] = x[c(2,3)] + v_phalf * timeStep
0.1	# v = .5 * (v_mhalf + v_phalf)
	# v_mhalf = v_phalf
	#"update densities, pressures, accelerations"
	#print("Starting rho calc")
1127.02	# rho = calculate_density(x, m, smoothingLength, rho, totalParticles, dimensions)
	#print("Starting p calc")
0.02	# P = presureConstant * rho^(1+1/PolyIndex)
	#("Starting accel calc")
3068.08	# accel = calculate_Acceleration(x, v, m, rho, P, damping, lambda, smoothingLength, accel, totalParticles, dimensions)
	#print("Starting save plot")
0.26	# png(file = paste("./OutputPlots/After", i,"loops.png", sep = ""))
0.54	# plot(x$xPos, x$yPos, xlim = c(-1, 3), ylim = c(-2, 2))
1.16	# dev.off()
0.2	# print(paste("Done loop", i, "at",Sys.time(), sep=" "))
	# }
	#})
	#RRprofStop()
	print(paste("Done at", Sys.time()))
	# print(profilingOutput)
	RRprofReport(file.name = "GUIProfiling_GradKernel.txt", reportname="GUIProfiling_GradKernel.html")
	#summaryRprof("Profiling_GradKernel.txt", lines="show")
	}