```
/*
COMPLETENESS DISCLAIMER: The source code for all file, graphic, string, and memory primitives
such as FileAddLine, Gcircle, StrDelimit, and MemClear used herein is not included but could easily be
created for the platform for which this source code will be compiled
*/

/*---------------------------------------------------------------------------------------------------------------------- */
/*                                                                                                                      */
/*                                                                                                                      */
/*                          BLACK HOLE INTERNAL DENSITY PROFILE                                                         */
/*                                                                                                                      */
/*                                                                                                                      */
/*---------------------------------------------------------------------------------------------------------------------- */
/* (c) Copyright Terence Witt 2007, All Rights Reserved */
/* NULLPHYSICS.COM */
/* BLACK HOLE DENSITY PROFILE */

/* master includes */
#include "witt.h"
#include "resource.h"
#include "zproject.h"

/* add a line to black hole density file */
void AddCoreLine(char *corefile, char tracechar, double r, double rhor)
{
        char    af[200];

        sprintf(af, "%c                                %4.3e        %4.3e", tracechar, r, rhor);
        FileAddLine(corefile, af);
}

/*---------------------------------------------------------------------------------------------------------------------- */
/*                                                                                                                      */
/*                                  BLACK HOLE DENSITY PROFILE                                                          */
/*                                                                                                                      */
/*---------------------------------------------------------------------------------------------------------------------- */
/* process density profile in black hole */
void ProcessBlackHole(char *corefile, double coremass, BOOL nonadiabatic)
{
        double  greso=100000, totmass=0.0, rhor=0.0, Rs, srat=0.0, dR=1.0, dV, dM, Rxc=0.0, resid=0.0, r=0.0;
        double  gpotfinal1=0.0, gpotfinal2=0.0, rfinal=0.0, RsHigh=0.0, masslimit, gdet;
        double  rhorini, rhornuke, rhorhyper, rhorgrad, rhorfinal=0.0, rhorprofile, percentdone;
        long    count=0, dither, centerpoints=100, ncount=0;
        char    af[200], tracechar;
        BOOL    first=TRUE;

        masslimit = coremass*Msun;
        /* mp is proton mass, Rp is proton core radius */
        rhornuke = mp/(12.0*Rp*Rp*Rp);
        rhorhyper = (6.0*mp)/(Rp*Rp*Rp);

        /* standard neutral nuclear density and hyperdensity */
        rhorini = rhornuke;
        tracechar = '1';

        /* schwarzschild limit */
        Rs = coremass*(2.0*G*Msun)/(c*c);

        /* search for distant surface for low-density objects */
        if(coremass<=4.0)
        {
                /* use 10% above a constant density radius */
```

```
		RsHigh = 1.2*pow(((3.0*masslimit)/(4.0*PI*rhorini)), 0.3333);
}
if((coremass>4.0) && (coremass<=6.0))
{
		RsHigh = 1.3*pow(((3.0*masslimit)/(4.0*PI*rhorini)), 0.3333);
}
if((coremass>6.0) && (coremass<=8.0))
{
		RsHigh = 1.4*pow(((3.0*masslimit)/(4.0*PI*rhorini)), 0.3333);
}
if((coremass>8.0) && (coremass<=10.0))
{
		RsHigh = 1.5*pow(((3.0*masslimit)/(4.0*PI*rhorini)), 0.3333);
}
/* assume galactic black hole at this mass */
if(coremass>=1000000.0)
{
		RsHigh = 1.1*Rs;
		/* nuclear density needs 32M resolution, 100 centerpoints;
		hyperdensity needs 600M, 2000 centerpoints */
		/* 600M, 2000 centerpoints does full density range for Milky Way's singularity */
		greso = 600000000;
		centerpoints = 2000;
}else{
		if(coremass>=100000.0)
		{
			RsHigh = 1.1*Rs;
			greso = 1000000;
		}else{
			if(coremass>10.0)
			{
				RsHigh = 1.1*Rs;
				greso = 100000;
			}
		}
}

/* if a baseline density gradient is used */
rhorgrad = (rhorhyper - rhornuke)/(RsHigh*RsHigh);

if(!corefile[0])
{
		PopWarning("Process Black Hole", "No Output File");
		return;
}

/* FIRST STEP ---------------------------------------------------------------------------------------- */
/* generate a density distribution in the absence of pressure */
/* density at a given location is the product of the density profile to that location */
Gframe(MESSAGEFRAME);
Gbold(1);
Gtextframesmall(MESSAGEFRAME, COLORBLUE, "", "Buiding Graph...", 24);
FileOutLine(corefile, "Trace		Scaling		Differential	x		y");

/* build a singularity a layer at a time */
/* use gravitational potential to expand layers and pressure to contract them */
/* keep dM the same in all layers */
dither = (long)(greso/MAXFXPOINTS);
r = dR;
dR = RsHigh/greso;
totmass = 0.0;
for(r=dR; r<RsHigh; r+=dR)
```

```
{
	/* test for real potential */
	gdet = (2.0*G*totmass)/(r*c*c);
	if(gdet<=1.0)
	{
		dV = 4.0*PI*r*r*dR;

		resid = sqrt(1.0 - gdet);

		/* try a baseline density which starts at hyperdense and falls to nuclear, based on r */
		rhorprofile = rhorhyper - (r*r*rhorgrad);

		if(nonadiabatic)
		{
			/* density based on gravitational potential, baseline nuclear, and energy loss */
			rhor = rhorprofile*(resid*resid*resid*resid);
		}else{
			/* density based on gravitational potential and baseline nuclear density only */
			rhor = rhorprofile*(resid*resid*resid);
		}

		dM = dV*rhor;
		totmass += dM;

		/* rewrite mass distribution each time, thickness starts at dR */
		if(count<centerpoints)
		{
			AddCoreLine(corefile, tracechar, r, rhor);
		}else{
			if(!(count%dither))
			{
				AddCoreLine(corefile, tracechar, r, rhor);
			}
		}
		count++;
	}else{
		if(first)
		{
			srat = r/Rs;
			first = FALSE;
		}
	}

	/* total mass has accumulated, exit */
	if(totmass>=masslimit)
	{
		rfinal = r;
		rhorfinal = rhor;
		break;
	}

	/* status and break */
	if(!(ncount++%1000000))
	{
		percentdone = 100.0*(r/RsHigh);
		sprintf(af, "Done: %4.2f %%", percentdone);
		Gtextframe(MESSAGEFRAME, COLORBLUE, "", af);
		if(KeyDown(VK_ESCAPE))
		{
			break;
		}
	}
```

```
	}
	ProcessFunction();

	/* calculate surface potential */
	if(rfinal>0.0)
	{
		gpotfinal1 = sqrt(1.0 - ((2.0*G*totmass)/(c*c*rfinal))) - 1.0;
	}
	if(rhorfinal>0.0)
	{
		gpotfinal2 = pow(rhorfinal/rhornuke, 0.33333) - 1.0;
		/* expanded particle radius defined purely by change in energy density */
		Rxc = pow((rhornuke/rhorfinal), 0.3333);
	}

	Gframe(MESSAGEFRAME);
	Gbold(1);
	sprintf(af, "M: %5.3f   R: %8.7e   Rxc: %5.3f   Sratio: %5.3f   Pot1/2: %7.6f/%7.6f",
		totmass/Msun, rfinal, Rxc, srat, gpotfinal1, gpotfinal2);
	Gtextframe(MESSAGEFRAME, COLORBLUE, "", af);
	Gbold(0);
}
```