AIMIS Final Code Listing

Alexis Enston 4026
The Kings School Chester 40317

May 5, 2015

Contents

1 ctlDirection.cs

```
   /* AIMIS
Copyright (C) 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace AIMIS
{
    public partial class ctlDirection : UserControl
    {
        //control for specifying angle in degrees, and showing a graphic of the direction

        public float fAngle;

        public ctlDirection()
        {
            InitializeComponent();
        }

        private void nmDirectionAngle_ValueChanged(object sender, EventArgs e)
        {
            if (nmDirectionAngle.Value == 360)
                nmDirectionAngle.Value = 0;
            if (nmDirectionAngle.Value == -1)
                nmDirectionAngle.Value = 350;

            fAngle = ((float)nmDirectionAngle.Value * (float)Math.PI) / 180;
            grpDirection.Refresh();
        }

        private void grpDirection_Paint(object sender, PaintEventArgs e)
        {
            Pen pen = new Pen(Color.Red);
            e.Graphics.DrawEllipse(pen, 70, 40, 100, 100);

            e.Graphics.DrawLine(pen, 120, 90, 120 + (float)Math.Cos(fAngle) * 50, 90 + (
                float)Math.Sin(fAngle) * 50);
        }
    }
}
```

2 ctlDirection.Designer.cs

```
namespace AIMIS
{
    partial class ctlDirection
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
            otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Component Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.grpDirection = new System.Windows.Forms.GroupBox();
            this.label5 = new System.Windows.Forms.Label();
            this.nmDirectionAngle = new System.Windows.Forms.NumericUpDown();
            this.grpDirection.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.nmDirectionAngle)).BeginInit
                ();
            this.SuspendLayout();
            //
            // grpDirection
            //
            this.grpDirection.Controls.Add(this.label5);
            this.grpDirection.Controls.Add(this.nmDirectionAngle);
            this.grpDirection.Location = new System.Drawing.Point(3, 3);
            this.grpDirection.Name = "grpDirection";
            this.grpDirection.Size = new System.Drawing.Size(205, 154);
            this.grpDirection.TabIndex = 11;
            this.grpDirection.TabStop = false;
            this.grpDirection.Text = "Direction";
            this.grpDirection.Paint += new System.Windows.Forms.PaintEventHandler(this.
                grpDirection_Paint);
            //
            // label5
            //
            this.label5.AutoSize = true;
            this.label5.Location = new System.Drawing.Point(14, 17);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(37, 13);
            this.label5.TabIndex = 1;
            this.label5.Text = "Angle:";
            //
            // nmDirectionAngle
            //
            this.nmDirectionAngle.Increment = new decimal(new int[] {
            10,
            0,
            0,
            0});
            this.nmDirectionAngle.Location = new System.Drawing.Point(57, 15);
            this.nmDirectionAngle.Maximum = new decimal(new int[] {
            360,
```

```
            0,
            0,
            0});
            this.nmDirectionAngle.Minimum = new decimal(new int[] {
            1,
            0,
            0,
            -2147483648});
            this.nmDirectionAngle.Name = "nmDirectionAngle";
            this.nmDirectionAngle.Size = new System.Drawing.Size(120, 20);
            this.nmDirectionAngle.TabIndex = 0;
            this.nmDirectionAngle.ValueChanged += new System.EventHandler(this.
                nmDirectionAngle_ValueChanged);
            //
            // ctlDirection
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.Controls.Add(this.grpDirection);
            this.Name = "ctlDirection";
            this.Size = new System.Drawing.Size(215, 163);
            this.grpDirection.ResumeLayout(false);
            this.grpDirection.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.nmDirectionAngle)).EndInit();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.GroupBox grpDirection;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.NumericUpDown nmDirectionAngle;
    }
}
```

3 frmAbout.cs

```
 using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace AIMIS
{
    public partial class frmAbout : Form
    {
        public frmAbout()
        {
            InitializeComponent();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

4 frmAbout.Designer.cs

```
namespace AIMIS
{
    partial class frmAbout
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
            otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new System.
                ComponentModel.ComponentResourceManager(typeof(frmAbout));
            this.label3 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.btnClose = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.SuspendLayout();
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Microsoft␣Sans␣Serif", 15.75F,
                System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((
                byte)(0)));
            this.label3.Location = new System.Drawing.Point(291, 9);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(104, 25);
            this.label3.TabIndex = 3;
            this.label3.Text = "AiMIS␣1.0";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(293, 43);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(241, 169);
            this.label4.TabIndex = 4;
            this.label4.Text = resources.GetString("label4.Text");
            //
            // pictureBox1
            //
            this.pictureBox1.Image = ((System.Drawing.Image)(resources.GetObject("
                pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(-5, -17);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(310, 299);
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
```

```
            // btnClose
            //
            this.btnClose.Image = global::AIMIS.Properties.Resources.dialog_close;
            this.btnClose.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnClose.Location = new System.Drawing.Point(468, 222);
            this.btnClose.Name = "btnClose";
            this.btnClose.Size = new System.Drawing.Size(66, 32);
            this.btnClose.TabIndex = 5;
            this.btnClose.Text = "Close";
            this.btnClose.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnClose.UseVisualStyleBackColor = true;
            this.btnClose.Click += new System.EventHandler(this.btnClose_Click);
            //
            // frmAbout
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(550, 266);
            this.Controls.Add(this.btnClose);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.pictureBox1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
            this.Name = "frmAbout";
            this.Text = "About␣AIMIS";
            this.TopMost = true;
            ((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Button btnClose;
    }
}
```

5 frmControl.cs

```
   /* AIMIS
Copyright (C) 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace AIMIS
{
    public partial class frmControl : Form
    {
        public gbVariables gbvars;
        public EventHandler ClearTrails;
        public tkui MainUIclass;
        public System.Threading.Thread thMainUI;
        private int SimSpeed = 0;

        public frmControl()
        {
            InitializeComponent();
        }

        private void rbNoTrails_CheckedChanged(object sender, EventArgs e)
        {
            gbvars.ShowTrails = !rbNoTrails.Checked;
            gbvars.ShortTrails = rbShortTrails.Checked;
        }

        private void btnClearTrails_Click(object sender, EventArgs e)
        {
            if(MainUIclass != null)
                MainUIclass.ClearTrails();

        }

        private void rkbSpeed_Scroll(object sender, EventArgs e)
        {
            if (MainUIclass != null)
                MainUIclass.SimulationSpeed = rkbSpeed.Value;

        }

        private void cboNewMass_SelectedValueChanged(object sender, EventArgs e)
        {

        }

        private void cboNewMass_TextChanged(object sender, EventArgs e)
        {
            float Mass = 0;
            //check to see if there is a valid number
            if(float.TryParse(cboNewMass.Text, out Mass )) {
                gbvars.NewObjectMass = Mass;
            }
            else
            {

                MessageBox.Show("Please␣enter␣a␣valid␣mass", "Invalid␣mass",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
        }

    }

    private void btnNewSim_Click(object sender, EventArgs e)
    {
        //don't start two sims, this will lead to problems with the ui
        if (thMainUI == null || !thMainUI.IsAlive)
        {

            //tkui class
            tkui TKUI = new tkui();
            TKUI.gbvars = gbvars;
            MainUIclass = TKUI;

            //setup thread
            thMainUI = new System.Threading.Thread(MainUIclass.Main);
            thMainUI.SetApartmentState(System.Threading.ApartmentState.STA);

            //load form for ui
            frmNewSim NewSimform = new frmNewSim();
            NewSimform.thMainUI = thMainUI;
            NewSimform.gbvars = gbvars;
            NewSimform.MainUIclass = MainUIclass;
            NewSimform.ShowDialog();
        }
        else
        {
            MessageBox.Show("Please␣close␣the␣current␣simulation␣\n␣before␣starting␣a␣
                new␣simulation.", "Unable␣to␣start␣simulation", MessageBoxButtons.OK,
                MessageBoxIcon.Warning);
        }

    }

    private void btnGraphs_Click(object sender, EventArgs e)
    {
        //load the graphs
        frmGraphs formgraph = new frmGraphs();
        formgraph.gbvars = gbvars;
        formgraph.Show();
    }

    private void frmControl_Load(object sender, EventArgs e)
    {

    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        //need something to save!
        if (MainUIclass == null)
            MessageBox.Show("Please␣start␣a␣simulation", "Error", MessageBoxButtons.OK
                , MessageBoxIcon.Error);
        else
        {
            if (MessageBox.Show("Clear␣trails␣when␣saving.␣This␣will␣result␣in␣a␣
                smaller␣file␣size.", "Clear␣trails?", MessageBoxButtons.YesNo,
                MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)
                MainUIclass.ClearTrails();
            //pause the simulation, and save it
            SimSpeed = MainUIclass.SimulationSpeed;
            MainUIclass.SimulationSpeed = 0;
            saveFileDialog1.ShowDialog();
        }
    }

    private void saveFileDialog1_FileOk(object sender, CancelEventArgs e)
    {
        MainUIclass.SavePlanets(saveFileDialog1.FileName);
        MainUIclass.SimulationSpeed = SimSpeed;
    }
```

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    gbvars.blFollowObject = ckDispObjToFollow.Checked;
}

private void nmObjToFollow_ValueChanged(object sender, EventArgs e)
{
    gbvars.intDispObToFollow = (int)nmObjToFollow.Value;
}

private void ckMoon_CheckedChanged(object sender, EventArgs e)
{
    gbvars.blAddMoon = ckMoon.Checked;
}

private int frmWidth;
private int frmHeight;
private bool blShowHide;

private void btnShowHide_Click(object sender, EventArgs e)
{
    if (blShowHide)
    {

        this.Width = frmWidth;
        this.Height = frmHeight;
        btnShowHide.Text = "Collapse";
        blShowHide = false;
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.
            FixedToolWindow;
    }
    else
    {
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
        frmWidth = this.Width;
        frmHeight = this.Height;
        this.Width = 94;
        this.Height = 26;
        btnShowHide.Text = "Expand";
        blShowHide = true;
    }
}

private void groupBox1_Enter(object sender, EventArgs e)
{

}

private void frmControl_FormClosing(object sender, FormClosingEventArgs e)
{
    if (thMainUI != null)
    {
        if (thMainUI.IsAlive)
            if (MessageBox.Show("Are you sure you want to quit the simulation?", "
                Terminate Simulation?", MessageBoxButtons.YesNo,MessageBoxIcon.
                Information) == System.Windows.Forms.DialogResult.Yes)
                thMainUI.Abort();
            else
                e.Cancel = true;


    }

}

private void chkAddObjAdvanced_CheckedChanged(object sender, EventArgs e)
{
    gbvars.blAddObjAdvanced = chkAddObjAdvanced.Checked;
}

private void btnAbout_Click(object sender, EventArgs e)
{
    frmAbout about = new frmAbout();
```

```
            about.ShowDialog();
        }

    }
}
```

6 frmControl.Designer.cs

```
namespace AIMIS
{
    partial class frmControl
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
            otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new System.
                ComponentModel.ComponentResourceManager(typeof(frmControl));
            this.grbTrails = new System.Windows.Forms.GroupBox();
            this.rbLongTrails = new System.Windows.Forms.RadioButton();
            this.rbShortTrails = new System.Windows.Forms.RadioButton();
            this.rbNoTrails = new System.Windows.Forms.RadioButton();
            this.rkbSpeed = new System.Windows.Forms.TrackBar();
            this.label1 = new System.Windows.Forms.Label();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.chkAddObjAdvanced = new System.Windows.Forms.CheckBox();
            this.ckMoon = new System.Windows.Forms.CheckBox();
            this.cboNewMass = new System.Windows.Forms.ComboBox();
            this.label2 = new System.Windows.Forms.Label();
            this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();
            this.ckDispObjToFollow = new System.Windows.Forms.CheckBox();
            this.nmObjToFollow = new System.Windows.Forms.NumericUpDown();
            this.btnAbout = new System.Windows.Forms.Button();
            this.btnShowHide = new System.Windows.Forms.Button();
            this.btnSave = new System.Windows.Forms.Button();
            this.btnGraphs = new System.Windows.Forms.Button();
            this.btnNewSim = new System.Windows.Forms.Button();
            this.btnClearTrails = new System.Windows.Forms.Button();
            this.grbTrails.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.rkbSpeed)).BeginInit();
            this.groupBox1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.nmObjToFollow)).BeginInit();
            this.SuspendLayout();
            //
            // grbTrails
            //
            this.grbTrails.Controls.Add(this.btnClearTrails);
            this.grbTrails.Controls.Add(this.rbLongTrails);
            this.grbTrails.Controls.Add(this.rbShortTrails);
            this.grbTrails.Controls.Add(this.rbNoTrails);
            this.grbTrails.Location = new System.Drawing.Point(12, 23);
            this.grbTrails.Name = "grbTrails";
            this.grbTrails.Size = new System.Drawing.Size(99, 151);
            this.grbTrails.TabIndex = 0;
            this.grbTrails.TabStop = false;
            this.grbTrails.Text = "Trails";
```

```
//
// rbLongTrails
//
this.rbLongTrails.AutoSize = true;
this.rbLongTrails.Location = new System.Drawing.Point(6, 65);
this.rbLongTrails.Name = "rbLongTrails";
this.rbLongTrails.Size = new System.Drawing.Size(73, 17);
this.rbLongTrails.TabIndex = 2;
this.rbLongTrails.Text = "Long␣trails";
this.rbLongTrails.UseVisualStyleBackColor = true;
this.rbLongTrails.CheckedChanged += new System.EventHandler(this.
    rbNoTrails_CheckedChanged);
//
// rbShortTrails
//
this.rbShortTrails.AutoSize = true;
this.rbShortTrails.Checked = true;
this.rbShortTrails.Location = new System.Drawing.Point(6, 42);
this.rbShortTrails.Name = "rbShortTrails";
this.rbShortTrails.Size = new System.Drawing.Size(74, 17);
this.rbShortTrails.TabIndex = 1;
this.rbShortTrails.TabStop = true;
this.rbShortTrails.Text = "Short␣trails";
this.rbShortTrails.UseVisualStyleBackColor = true;
this.rbShortTrails.CheckedChanged += new System.EventHandler(this.
    rbNoTrails_CheckedChanged);
//
// rbNoTrails
//
this.rbNoTrails.AutoSize = true;
this.rbNoTrails.Location = new System.Drawing.Point(6, 19);
this.rbNoTrails.Name = "rbNoTrails";
this.rbNoTrails.Size = new System.Drawing.Size(63, 17);
this.rbNoTrails.TabIndex = 0;
this.rbNoTrails.Text = "No␣trails";
this.rbNoTrails.UseVisualStyleBackColor = true;
this.rbNoTrails.CheckedChanged += new System.EventHandler(this.
    rbNoTrails_CheckedChanged);
//
// rkbSpeed
//
this.rkbSpeed.Location = new System.Drawing.Point(15, 193);
this.rkbSpeed.Maximum = 100;
this.rkbSpeed.Name = "rkbSpeed";
this.rkbSpeed.Size = new System.Drawing.Size(310, 45);
this.rkbSpeed.TabIndex = 1;
this.rkbSpeed.Value = 20;
this.rkbSpeed.Scroll += new System.EventHandler(this.rkbSpeed_Scroll);
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(12, 177);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(99, 13);
this.label1.TabIndex = 2;
this.label1.Text = "Speed␣of␣simulation";
//
// groupBox1
//
this.groupBox1.Controls.Add(this.chkAddObjAdvanced);
this.groupBox1.Controls.Add(this.ckMoon);
this.groupBox1.Controls.Add(this.cboNewMass);
this.groupBox1.Controls.Add(this.label2);
this.groupBox1.Location = new System.Drawing.Point(117, 23);
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(208, 75);
this.groupBox1.TabIndex = 3;
this.groupBox1.TabStop = false;
this.groupBox1.Text = "New␣Objects";
this.groupBox1.Enter += new System.EventHandler(this.groupBox1_Enter);
//
// chkAddObjAdvanced
```

```
            //
            this.chkAddObjAdvanced.AutoSize = true;
            this.chkAddObjAdvanced.Location = new System.Drawing.Point(9, 45);
            this.chkAddObjAdvanced.Name = "chkAddObjAdvanced";
            this.chkAddObjAdvanced.Size = new System.Drawing.Size(114, 17);
            this.chkAddObjAdvanced.TabIndex = 3;
            this.chkAddObjAdvanced.Text = "Advanced␣Options";
            this.chkAddObjAdvanced.UseVisualStyleBackColor = true;
            this.chkAddObjAdvanced.CheckedChanged += new System.EventHandler(this.
                chkAddObjAdvanced_CheckedChanged);
            //
            // ckMoon
            //
            this.ckMoon.AutoSize = true;
            this.ckMoon.Location = new System.Drawing.Point(127, 45);
            this.ckMoon.Name = "ckMoon";
            this.ckMoon.Size = new System.Drawing.Size(75, 17);
            this.ckMoon.TabIndex = 2;
            this.ckMoon.Text = "Add␣Moon";
            this.ckMoon.UseVisualStyleBackColor = true;
            this.ckMoon.CheckedChanged += new System.EventHandler(this.
                ckMoon_CheckedChanged);
            //
            // cboNewMass
            //
            this.cboNewMass.FormattingEnabled = true;
            this.cboNewMass.Items.AddRange(new object[] {
            "1",
            "5",
            "10",
            "25",
            "50",
            "100"});
            this.cboNewMass.Location = new System.Drawing.Point(56, 18);
            this.cboNewMass.Name = "cboNewMass";
            this.cboNewMass.Size = new System.Drawing.Size(146, 21);
            this.cboNewMass.TabIndex = 1;
            this.cboNewMass.Text = "5";
            this.cboNewMass.SelectedValueChanged += new System.EventHandler(this.
                cboNewMass_SelectedValueChanged);
            this.cboNewMass.TextChanged += new System.EventHandler(this.
                cboNewMass_TextChanged);
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(15, 21);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(35, 13);
            this.label2.TabIndex = 0;
            this.label2.Text = "Mass:";
            //
            // saveFileDialog1
            //
            this.saveFileDialog1.DefaultExt = "xml";
            this.saveFileDialog1.Filter = "XML␣file␣|*.xml";
            this.saveFileDialog1.FileOk += new System.ComponentModel.CancelEventHandler(
                this.saveFileDialog1_FileOk);
            //
            // ckDispObjToFollow
            //
            this.ckDispObjToFollow.AutoSize = true;
            this.ckDispObjToFollow.Location = new System.Drawing.Point(117, 105);
            this.ckDispObjToFollow.Name = "ckDispObjToFollow";
            this.ckDispObjToFollow.Size = new System.Drawing.Size(90, 17);
            this.ckDispObjToFollow.TabIndex = 7;
            this.ckDispObjToFollow.Text = "Follow␣Object";
            this.ckDispObjToFollow.UseVisualStyleBackColor = true;
            this.ckDispObjToFollow.CheckedChanged += new System.EventHandler(this.
                checkBox1_CheckedChanged);
            //
            // nmObjToFollow
            //
```

```
this.nmObjToFollow.Location = new System.Drawing.Point(213, 104);
this.nmObjToFollow.Name = "nmObjToFollow";
this.nmObjToFollow.Size = new System.Drawing.Size(112, 20);
this.nmObjToFollow.TabIndex = 8;
this.nmObjToFollow.ValueChanged += new System.EventHandler(this.
    nmObjToFollow_ValueChanged);
//
// btnAbout
//
this.btnAbout.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnAbout.Location = new System.Drawing.Point(279, 1);
this.btnAbout.Name = "btnAbout";
this.btnAbout.Size = new System.Drawing.Size(46, 23);
this.btnAbout.TabIndex = 10;
this.btnAbout.Text = "About";
this.btnAbout.UseVisualStyleBackColor = true;
this.btnAbout.Click += new System.EventHandler(this.btnAbout_Click);
//
// btnShowHide
//
this.btnShowHide.Image = global::AIMIS.Properties.Resources.view_restore;
this.btnShowHide.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnShowHide.Location = new System.Drawing.Point(2, 1);
this.btnShowHide.Name = "btnShowHide";
this.btnShowHide.Size = new System.Drawing.Size(90, 23);
this.btnShowHide.TabIndex = 9;
this.btnShowHide.Text = "Collapse";
this.btnShowHide.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnShowHide.UseVisualStyleBackColor = true;
this.btnShowHide.Click += new System.EventHandler(this.btnShowHide_Click);
//
// btnSave
//
this.btnSave.Image = global::AIMIS.Properties.Resources.document_save;
this.btnSave.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnSave.Location = new System.Drawing.Point(117, 164);
this.btnSave.Name = "btnSave";
this.btnSave.Size = new System.Drawing.Size(123, 28);
this.btnSave.TabIndex = 6;
this.btnSave.Text = "Save␣Simulation";
this.btnSave.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnSave.UseVisualStyleBackColor = true;
this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
//
// btnGraphs
//
this.btnGraphs.Image = global::AIMIS.Properties.Resources.
    office_chart_area_stacked;
this.btnGraphs.ImageAlign = System.Drawing.ContentAlignment.TopCenter;
this.btnGraphs.Location = new System.Drawing.Point(246, 128);
this.btnGraphs.Name = "btnGraphs";
this.btnGraphs.Size = new System.Drawing.Size(79, 64);
this.btnGraphs.TabIndex = 5;
this.btnGraphs.Text = "View␣Graphs";
this.btnGraphs.TextAlign = System.Drawing.ContentAlignment.BottomCenter;
this.btnGraphs.UseVisualStyleBackColor = true;
this.btnGraphs.Click += new System.EventHandler(this.btnGraphs_Click);
//
// btnNewSim
//
this.btnNewSim.Image = global::AIMIS.Properties.Resources.window_new;
this.btnNewSim.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnNewSim.Location = new System.Drawing.Point(117, 128);
this.btnNewSim.Name = "btnNewSim";
this.btnNewSim.Size = new System.Drawing.Size(123, 30);
this.btnNewSim.TabIndex = 4;
this.btnNewSim.Text = "New␣Simulation";
this.btnNewSim.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnNewSim.UseVisualStyleBackColor = true;
this.btnNewSim.Click += new System.EventHandler(this.btnNewSim_Click);
//
// btnClearTrails
//
```

```
            this.btnClearTrails.Image = global::AIMIS.Properties.Resources.edit_clear;
            this.btnClearTrails.ImageAlign = System.Drawing.ContentAlignment.TopCenter;
            this.btnClearTrails.Location = new System.Drawing.Point(6, 88);
            this.btnClearTrails.Name = "btnClearTrails";
            this.btnClearTrails.Size = new System.Drawing.Size(84, 48);
            this.btnClearTrails.TabIndex = 3;
            this.btnClearTrails.Text = "Clear␣Trails";
            this.btnClearTrails.TextAlign = System.Drawing.ContentAlignment.BottomCenter;
            this.btnClearTrails.UseVisualStyleBackColor = true;
            this.btnClearTrails.Click += new System.EventHandler(this.btnClearTrails_Click
                );
            //
            // frmControl
            //
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.None;
            this.ClientSize = new System.Drawing.Size(330, 229);
            this.Controls.Add(this.btnAbout);
            this.Controls.Add(this.btnShowHide);
            this.Controls.Add(this.nmObjToFollow);
            this.Controls.Add(this.ckDispObjToFollow);
            this.Controls.Add(this.btnSave);
            this.Controls.Add(this.btnGraphs);
            this.Controls.Add(this.btnNewSim);
            this.Controls.Add(this.groupBox1);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.rkbSpeed);
            this.Controls.Add(this.grbTrails);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
            this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.Name = "frmControl";
            this.Text = "AIMIS␣control␣window";
            this.TopMost = true;
            this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.
                frmControl_FormClosing);
            this.Load += new System.EventHandler(this.frmControl_Load);
            this.grbTrails.ResumeLayout(false);
            this.grbTrails.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.rkbSpeed)).EndInit();
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.nmObjToFollow)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.GroupBox grbTrails;
        private System.Windows.Forms.RadioButton rbLongTrails;
        private System.Windows.Forms.RadioButton rbShortTrails;
        private System.Windows.Forms.RadioButton rbNoTrails;
        private System.Windows.Forms.Button btnClearTrails;
        private System.Windows.Forms.TrackBar rkbSpeed;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.ComboBox cboNewMass;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button btnNewSim;
        private System.Windows.Forms.Button btnGraphs;
        private System.Windows.Forms.Button btnSave;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
        private System.Windows.Forms.CheckBox ckDispObjToFollow;
        private System.Windows.Forms.NumericUpDown nmObjToFollow;
        private System.Windows.Forms.CheckBox ckMoon;
        private System.Windows.Forms.Button btnShowHide;
        private System.Windows.Forms.CheckBox chkAddObjAdvanced;
        private System.Windows.Forms.Button btnAbout;
    }
}
```

7 frmEarthOrbit.cs

```
   /* AIMIS
Copyright (C) 2014, 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace AIMIS
{
    public partial class frmEarthOrbit : Form
    {
        public gbVariables gbvars;
        public tkui MainUIclass;
        public System.Threading.Thread thMainUI;

        public frmEarthOrbit()
        {
            InitializeComponent();
        }

        private void btnLaunch_Click(object sender, EventArgs e)
        {
            //add an object into orbit, based on the given parameters
            MainUIclass.NewPlanet(5f, 0f, (float)nmHeight.Value + MainUIclass.lstPlanets
                [0].Radius,
                (float)Math.Cos(-ctlDirection1.fAngle) * (float)nmSpeed.Value / 50,
                (float)Math.Sin(-ctlDirection1.fAngle) * (float)nmSpeed.Value / 50);

            MainUIclass.blShowGeostatDot = chPoint.Checked;
            if (chPoint.Checked)
            {
                MainUIclass.fAngleGeostat = MainUIclass.lstPlanets[0].RotationAngle;
            }
        }

        private void frmEarthOrbit_Load(object sender, EventArgs e)
        {

        }

        private void btnClear_Click(object sender, EventArgs e)
        {
            MainUIclass.lstPlanets.RemoveRange(1, MainUIclass.lstPlanets.Count - 1);
            MainUIclass.ClearTrails();
        }
    }
}
```

8 frmEarthOrbit.Designer.cs

```
namespace AIMIS
{
    partial class frmEarthOrbit
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
            otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.nmHeight = new System.Windows.Forms.NumericUpDown();
            this.nmSpeed = new System.Windows.Forms.NumericUpDown();
            this.btnClear = new System.Windows.Forms.Button();
            this.ctlDirection1 = new AIMIS.ctlDirection();
            this.chPoint = new System.Windows.Forms.CheckBox();
            this.btnLaunch = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.nmHeight)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmSpeed)).BeginInit();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(12, 14);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(112, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Height above surface:";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(83, 40);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(41, 13);
            this.label2.TabIndex = 1;
            this.label2.Text = "Speed:";
            //
            // nmHeight
            //
            this.nmHeight.DecimalPlaces = 2;
            this.nmHeight.Increment = new decimal(new int[] {
            2,
            0,
            0,
            65536});
            this.nmHeight.Location = new System.Drawing.Point(130, 12);
            this.nmHeight.Name = "nmHeight";
```

```
this.nmHeight.Size = new System.Drawing.Size(114, 20);
this.nmHeight.TabIndex = 3;
//
// nmSpeed
//
this.nmSpeed.DecimalPlaces = 2;
this.nmSpeed.Increment = new decimal(new int[] {
1,
0,
0,
131072});
this.nmSpeed.Location = new System.Drawing.Point(130, 38);
this.nmSpeed.Name = "nmSpeed";
this.nmSpeed.Size = new System.Drawing.Size(114, 20);
this.nmSpeed.TabIndex = 4;
//
// btnClear
//
this.btnClear.Image = global::AIMIS.Properties.Resources.edit_clear;
this.btnClear.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnClear.Location = new System.Drawing.Point(88, 252);
this.btnClear.Name = "btnClear";
this.btnClear.Size = new System.Drawing.Size(75, 34);
this.btnClear.TabIndex = 8;
this.btnClear.Text = "Clear";
this.btnClear.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnClear.UseVisualStyleBackColor = true;
this.btnClear.Click += new System.EventHandler(this.btnClear_Click);
//
// ctlDirection1
//
this.ctlDirection1.Location = new System.Drawing.Point(35, 64);
this.ctlDirection1.Name = "ctlDirection1";
this.ctlDirection1.Size = new System.Drawing.Size(215, 163);
this.ctlDirection1.TabIndex = 7;
//
// chPoint
//
this.chPoint.AutoSize = true;
this.chPoint.Location = new System.Drawing.Point(35, 229);
this.chPoint.Name = "chPoint";
this.chPoint.Size = new System.Drawing.Size(135, 17);
this.chPoint.TabIndex = 9;
this.chPoint.Text = "Show␣Point␣on␣Surface";
this.chPoint.UseVisualStyleBackColor = true;
//
// btnLaunch
//
this.btnLaunch.Image = global::AIMIS.Properties.Resources.fork;
this.btnLaunch.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnLaunch.Location = new System.Drawing.Point(169, 252);
this.btnLaunch.Name = "btnLaunch";
this.btnLaunch.Size = new System.Drawing.Size(75, 34);
this.btnLaunch.TabIndex = 6;
this.btnLaunch.Text = "Launch";
this.btnLaunch.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnLaunch.UseVisualStyleBackColor = true;
this.btnLaunch.Click += new System.EventHandler(this.btnLaunch_Click);
//
// frmEarthOrbit
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(263, 298);
this.Controls.Add(this.chPoint);
this.Controls.Add(this.btnClear);
this.Controls.Add(this.ctlDirection1);
this.Controls.Add(this.btnLaunch);
this.Controls.Add(this.nmSpeed);
this.Controls.Add(this.nmHeight);
this.Controls.Add(this.label2);
this.Controls.Add(this.label1);
this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
```

```
            this.Name = "frmEarthOrbit";
            this.Text = "Orbit␣Simulator";
            this.TopMost = true;
            this.Load += new System.EventHandler(this.frmEarthOrbit_Load);
            ((System.ComponentModel.ISupportInitialize)(this.nmHeight)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmSpeed)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.NumericUpDown nmHeight;
        private System.Windows.Forms.NumericUpDown nmSpeed;
        private System.Windows.Forms.Button btnLaunch;
        private ctlDirection ctlDirection1;
        private System.Windows.Forms.Button btnClear;
        private System.Windows.Forms.CheckBox chPoint;
    }
}
```

9 frmGraphs.cs

```
   /* AIMIS
Copyright (C) 2014, 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace AIMIS
{
    public partial class frmGraphs : Form
    {
        public gbVariables gbvars;

        public frmGraphs()
        {
            InitializeComponent();
        }

        private void chart1_Click(object sender, EventArgs e)
        {

        }

        private void frmGraphs_Load(object sender, EventArgs e)
        {
            List<float> velocit = new List<float>(gbvars.lstVelocities);

            //setup the graph
            chart1.DataSource = velocit;
                chart1.DataBind();
                chart1.Series[0].IsXValueIndexed = true;
                chart1.Series.First().YValueMembers = "X";
                chart1.Update();
        }


        private void button1_Click(object sender, EventArgs e)
        {
            gbvars.lstVelocities.Clear();
        }

        private void timerUpdate_Tick(object sender, EventArgs e)
        {
            //update the graph every x seconds
            List<float> velocit = new List<float>(gbvars.lstVelocities);

            nudPlanetIndex.Value = gbvars.intObjectToTrack;

            chart1.DataSource = velocit;

            chart1.DataBind();
            chart1.Update();
        }



        private void nudPlanetIndex_ValueChanged(object sender, EventArgs e)
        {
            gbvars.intObjectToTrack = (int)nudPlanetIndex.Value;
```

```
            gbvars.lstVelocities.Clear();

        }

        private void chTrackNew_CheckedChanged(object sender, EventArgs e)
        {
            gbvars.blTrackNewObject = chTrackNew.Checked;
        }

        private void chTrackObject_CheckedChanged(object sender, EventArgs e)
        {
            gbvars.blGraphTrack = chTrackObject.Checked;
        }


    }
}
```

10 frmGraphs.Designer.cs

```
namespace AIMIS
{
    partial class frmGraphs
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
            otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 = new
                System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Series series1 = new System.
                Windows.Forms.DataVisualization.Charting.Series();
            System.ComponentModel.ComponentResourceManager resources = new System.
                ComponentModel.ComponentResourceManager(typeof(frmGraphs));
            this.chart1 = new System.Windows.Forms.DataVisualization.Charting.Chart();
            this.timerUpdate = new System.Windows.Forms.Timer(this.components);
            this.nudPlanetIndex = new System.Windows.Forms.NumericUpDown();
            this.label1 = new System.Windows.Forms.Label();
            this.chTrackNew = new System.Windows.Forms.CheckBox();
            this.chTrackObject = new System.Windows.Forms.CheckBox();
            this.btnClear = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.nudPlanetIndex)).BeginInit();
            this.SuspendLayout();
            //
            // chart1
            //
            this.chart1.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.
                Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            chartArea1.Name = "ChartArea1";
            this.chart1.ChartAreas.Add(chartArea1);
            this.chart1.Location = new System.Drawing.Point(2, 2);
            this.chart1.Name = "chart1";
            series1.ChartArea = "ChartArea1";
            series1.ChartType = System.Windows.Forms.DataVisualization.Charting.
                SeriesChartType.Spline;
            series1.MarkerBorderWidth = 2;
            series1.Name = "Series1";
            series1.YValuesPerPoint = 6;
            this.chart1.Series.Add(series1);
            this.chart1.Size = new System.Drawing.Size(638, 330);
            this.chart1.TabIndex = 0;
            this.chart1.Text = "chart1";
            this.chart1.Click += new System.EventHandler(this.chart1_Click);
            //
```

```
            // timerUpdate
            //
            this.timerUpdate.Enabled = true;
            this.timerUpdate.Interval = 500;
            this.timerUpdate.Tick += new System.EventHandler(this.timerUpdate_Tick);
            //
            // nudPlanetIndex
            //
            this.nudPlanetIndex.Anchor = ((System.Windows.Forms.AnchorStyles)((System.
                Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left
                )));
            this.nudPlanetIndex.Location = new System.Drawing.Point(107, 359);
            this.nudPlanetIndex.Name = "nudPlanetIndex";
            this.nudPlanetIndex.Size = new System.Drawing.Size(120, 20);
            this.nudPlanetIndex.TabIndex = 3;
            this.nudPlanetIndex.ValueChanged += new System.EventHandler(this.
                nudPlanetIndex_ValueChanged);
            //
            // label1
            //
            this.label1.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left)));
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(22, 361);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(79, 13);
            this.label1.TabIndex = 4;
            this.label1.Text = "Planet to track:";
            //
            // chTrackNew
            //
            this.chTrackNew.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left)));
            this.chTrackNew.AutoSize = true;
            this.chTrackNew.Checked = true;
            this.chTrackNew.CheckState = System.Windows.Forms.CheckState.Checked;
            this.chTrackNew.Location = new System.Drawing.Point(350, 362);
            this.chTrackNew.Name = "chTrackNew";
            this.chTrackNew.Size = new System.Drawing.Size(114, 17);
            this.chTrackNew.TabIndex = 5;
            this.chTrackNew.Text = "Track new objects";
            this.chTrackNew.UseVisualStyleBackColor = true;
            this.chTrackNew.CheckedChanged += new System.EventHandler(this.
                chTrackNew_CheckedChanged);
            //
            // chTrackObject
            //
            this.chTrackObject.Anchor = ((System.Windows.Forms.AnchorStyles)((System.
                Windows.Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left
                )));
            this.chTrackObject.AutoSize = true;
            this.chTrackObject.Location = new System.Drawing.Point(470, 362);
            this.chTrackObject.Name = "chTrackObject";
            this.chTrackObject.Size = new System.Drawing.Size(99, 17);
            this.chTrackObject.TabIndex = 6;
            this.chTrackObject.Text = "Highlight object";
            this.chTrackObject.UseVisualStyleBackColor = true;
            this.chTrackObject.CheckedChanged += new System.EventHandler(this.
                chTrackObject_CheckedChanged);
            //
            // btnClear
            //
            this.btnClear.Anchor = ((System.Windows.Forms.AnchorStyles)((System.Windows.
                Forms.AnchorStyles.Bottom | System.Windows.Forms.AnchorStyles.Left)));
            this.btnClear.Image = global::AIMIS.Properties.Resources.edit_clear;
            this.btnClear.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnClear.Location = new System.Drawing.Point(233, 352);
            this.btnClear.Name = "btnClear";
            this.btnClear.Size = new System.Drawing.Size(99, 32);
            this.btnClear.TabIndex = 2;
            this.btnClear.Text = "Clear Graph";
            this.btnClear.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnClear.UseVisualStyleBackColor = true;
```

```
            this.btnClear.Click += new System.EventHandler(this.button1_Click);
            //
            // frmGraphs
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(640, 396);
            this.Controls.Add(this.chTrackObject);
            this.Controls.Add(this.chTrackNew);
            this.Controls.Add(this.chart1);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.nudPlanetIndex);
            this.Controls.Add(this.btnClear);
            this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.Name = "frmGraphs";
            this.Text = "AIMIS Graphs";
            this.Load += new System.EventHandler(this.frmGraphs_Load);
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.nudPlanetIndex)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        private System.Windows.Forms.Button btnClear;
        private System.Windows.Forms.Timer timerUpdate;
        private System.Windows.Forms.NumericUpDown nudPlanetIndex;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.CheckBox chTrackNew;
        private System.Windows.Forms.CheckBox chTrackObject;
    }
}
```

11 frmNewObjAdv.cs

```
   /* AIMIS
Copyright (C) 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace AIMIS
{
    public partial class frmNewObjAdv : Form
    {
        public float fAngle = 0;
        public gbVariables gbvars;
        public float fMass;
        public bool blAddMoon;
        public float fMoonMass;
        public float fMoonDistance;
        public float fSpeed;
        public string stTextureFilename;
        public float fRotation;
        public bool blFixed;

        public frmNewObjAdv()
        {
            InitializeComponent();

        }


        private void frmNewObjAdv_Load(object sender, EventArgs e)
        {
            nmMass.Value = (decimal)gbvars.NewObjectMass;
            ckAddMoon.Checked = gbvars.blAddMoon;
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            //store the variables so they can be accessed by tkui
            fAngle = ctlDirection1.fAngle;
            fMass = (float)nmMass.Value;
            fMoonMass = (float)nmMoonMass.Value;
            fMoonDistance = (float)nmMoonRadius.Value;
            blAddMoon = ckAddMoon.Checked;
            fSpeed = (float)nmSpeed.Value / 1000f;
            stTextureFilename = txtTexture.Text;
            fRotation = (float)nmRotation.Value;
            blFixed = chFixed.Checked;

            this.DialogResult = System.Windows.Forms.DialogResult.OK;
            this.Close();
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void nmMass_ValueChanged(object sender, EventArgs e)
        {
```

```
            //calculate radius
            lbRadius.Text = "(Radius␣of␣main␣planet␣is␣" + Decimal.Round((decimal)Math.Pow
                (((double)nmMass.Value * 3) / (Math.PI * 4 * 8000), (double)1 / 3), 2).
                ToString() + "␣)";

        }

        private void btnLoadTexture_Click(object sender, EventArgs e)
        {
            openFileDialog1.ShowDialog();
        }

        private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
        {
            txtTexture.Text = openFileDialog1.FileName;
        }
    }
}
```

12 frmNewObjAdv.Designer.cs

```
namespace AIMIS
{
    partial class frmNewObjAdv
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
            otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.label1 = new System.Windows.Forms.Label();
            this.nmMass = new System.Windows.Forms.NumericUpDown();
            this.ckAddMoon = new System.Windows.Forms.CheckBox();
            this.label2 = new System.Windows.Forms.Label();
            this.nmMoonMass = new System.Windows.Forms.NumericUpDown();
            this.label3 = new System.Windows.Forms.Label();
            this.nmMoonRadius = new System.Windows.Forms.NumericUpDown();
            this.grpMoon = new System.Windows.Forms.GroupBox();
            this.lbRadius = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.nmSpeed = new System.Windows.Forms.NumericUpDown();
            this.label9 = new System.Windows.Forms.Label();
            this.txtTexture = new System.Windows.Forms.TextBox();
            this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
            this.label10 = new System.Windows.Forms.Label();
            this.nmRotation = new System.Windows.Forms.NumericUpDown();
            this.chFixed = new System.Windows.Forms.CheckBox();
            this.btnLoadTexture = new System.Windows.Forms.Button();
            this.ctlDirection1 = new AIMIS.ctlDirection();
            this.btnCancel = new System.Windows.Forms.Button();
            this.btnOK = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.nmMass)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmMoonMass)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmMoonRadius)).BeginInit();
            this.grpMoon.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.nmSpeed)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmRotation)).BeginInit();
            this.SuspendLayout();
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(40, 17);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(35, 13);
            this.label1.TabIndex = 0;
            this.label1.Text = "Mass:";
            //
            // nmMass
            //
```

```
this.nmMass.Location = new System.Drawing.Point(81, 15);
this.nmMass.Maximum = new decimal(new int[] {
10000,
0,
0,
0});
this.nmMass.Name = "nmMass";
this.nmMass.Size = new System.Drawing.Size(136, 20);
this.nmMass.TabIndex = 1;
this.nmMass.ValueChanged += new System.EventHandler(this.nmMass_ValueChanged);
//
// ckAddMoon
//
this.ckAddMoon.AutoSize = true;
this.ckAddMoon.Location = new System.Drawing.Point(99, 14);
this.ckAddMoon.Name = "ckAddMoon";
this.ckAddMoon.Size = new System.Drawing.Size(73, 17);
this.ckAddMoon.TabIndex = 2;
this.ckAddMoon.Text = "add␣moon";
this.ckAddMoon.UseVisualStyleBackColor = true;
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(14, 39);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(77, 13);
this.label2.TabIndex = 3;
this.label2.Text = "Mass␣of␣Moon:";
//
// nmMoonMass
//
this.nmMoonMass.Location = new System.Drawing.Point(99, 37);
this.nmMoonMass.Name = "nmMoonMass";
this.nmMoonMass.Size = new System.Drawing.Size(120, 20);
this.nmMoonMass.TabIndex = 4;
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(14, 65);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(76, 13);
this.label3.TabIndex = 5;
this.label3.Text = "Orbital␣Radius:";
//
// nmMoonRadius
//
this.nmMoonRadius.DecimalPlaces = 2;
this.nmMoonRadius.Increment = new decimal(new int[] {
1,
0,
0,
131072});
this.nmMoonRadius.Location = new System.Drawing.Point(99, 63);
this.nmMoonRadius.Name = "nmMoonRadius";
this.nmMoonRadius.Size = new System.Drawing.Size(120, 20);
this.nmMoonRadius.TabIndex = 6;
//
// grpMoon
//
this.grpMoon.Controls.Add(this.lbRadius);
this.grpMoon.Controls.Add(this.label2);
this.grpMoon.Controls.Add(this.nmMoonRadius);
this.grpMoon.Controls.Add(this.ckAddMoon);
this.grpMoon.Controls.Add(this.label3);
this.grpMoon.Controls.Add(this.nmMoonMass);
this.grpMoon.Location = new System.Drawing.Point(1, 142);
this.grpMoon.Name = "grpMoon";
this.grpMoon.Size = new System.Drawing.Size(235, 103);
this.grpMoon.TabIndex = 7;
this.grpMoon.TabStop = false;
this.grpMoon.Text = "Moon";
```

```
//
// lbRadius
//
this.lbRadius.AutoSize = true;
this.lbRadius.Location = new System.Drawing.Point(14, 84);
this.lbRadius.Name = "lbRadius";
this.lbRadius.Size = new System.Drawing.Size(117, 13);
this.lbRadius.TabIndex = 7;
this.lbRadius.Text = "(Radius␣of␣planet␣is␣XX)";
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(9, 43);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(68, 13);
this.label4.TabIndex = 8;
this.label4.Text = "Initial␣Speed:";
//
// nmSpeed
//
this.nmSpeed.DecimalPlaces = 2;
this.nmSpeed.Increment = new decimal(new int[] {
5,
0,
0,
65536});
this.nmSpeed.Location = new System.Drawing.Point(81, 41);
this.nmSpeed.Name = "nmSpeed";
this.nmSpeed.Size = new System.Drawing.Size(136, 20);
this.nmSpeed.TabIndex = 9;
this.nmSpeed.Value = new decimal(new int[] {
5,
0,
0,
0});
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(29, 70);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(46, 13);
this.label9.TabIndex = 14;
this.label9.Text = "Texture:";
//
// txtTexture
//
this.txtTexture.Location = new System.Drawing.Point(81, 67);
this.txtTexture.Name = "txtTexture";
this.txtTexture.Size = new System.Drawing.Size(108, 20);
this.txtTexture.TabIndex = 15;
//
// openFileDialog1
//
this.openFileDialog1.Filter = "PNG␣images|*.png|JPEG␣Image|*.jpg";
this.openFileDialog1.FileOk += new System.ComponentModel.CancelEventHandler(
    this.openFileDialog1_FileOk);
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(27, 95);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(50, 13);
this.label10.TabIndex = 17;
this.label10.Text = "Rotation:";
//
// nmRotation
//
this.nmRotation.DecimalPlaces = 2;
this.nmRotation.Increment = new decimal(new int[] {
1,
```

```
0,
0,
131072});
this.nmRotation.Location = new System.Drawing.Point(81, 93);
this.nmRotation.Name = "nmRotation";
this.nmRotation.Size = new System.Drawing.Size(136, 20);
this.nmRotation.TabIndex = 18;
//
// chFixed
//
this.chFixed.AutoSize = true;
this.chFixed.Location = new System.Drawing.Point(81, 119);
this.chFixed.Name = "chFixed";
this.chFixed.Size = new System.Drawing.Size(83, 17);
this.chFixed.TabIndex = 2;
this.chFixed.Text = "Fixed␣object";
this.chFixed.UseVisualStyleBackColor = true;
//
// btnLoadTexture
//
this.btnLoadTexture.Location = new System.Drawing.Point(192, 67);
this.btnLoadTexture.Name = "btnLoadTexture";
this.btnLoadTexture.Size = new System.Drawing.Size(25, 20);
this.btnLoadTexture.TabIndex = 16;
this.btnLoadTexture.Text = "...";
this.btnLoadTexture.UseVisualStyleBackColor = true;
this.btnLoadTexture.Click += new System.EventHandler(this.btnLoadTexture_Click
    );
//
// ctlDirection1
//
this.ctlDirection1.Location = new System.Drawing.Point(242, 12);
this.ctlDirection1.Name = "ctlDirection1";
this.ctlDirection1.Size = new System.Drawing.Size(215, 163);
this.ctlDirection1.TabIndex = 19;
//
// btnCancel
//
this.btnCancel.Image = global::AIMIS.Properties.Resources.dialog_close;
this.btnCancel.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnCancel.Location = new System.Drawing.Point(270, 211);
this.btnCancel.Name = "btnCancel";
this.btnCancel.Size = new System.Drawing.Size(73, 34);
this.btnCancel.TabIndex = 12;
this.btnCancel.Text = "Cancel";
this.btnCancel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnCancel.UseVisualStyleBackColor = true;
this.btnCancel.Click += new System.EventHandler(this.btnCancel_Click);
//
// btnOK
//
this.btnOK.Image = global::AIMIS.Properties.Resources.list_add;
this.btnOK.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnOK.Location = new System.Drawing.Point(349, 211);
this.btnOK.Name = "btnOK";
this.btnOK.Size = new System.Drawing.Size(97, 34);
this.btnOK.TabIndex = 11;
this.btnOK.Text = "Add␣Object";
this.btnOK.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnOK.UseVisualStyleBackColor = true;
this.btnOK.Click += new System.EventHandler(this.btnOK_Click);
//
// frmNewObjAdv
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(458, 261);
this.Controls.Add(this.ctlDirection1);
this.Controls.Add(this.chFixed);
this.Controls.Add(this.nmRotation);
this.Controls.Add(this.label10);
this.Controls.Add(this.btnLoadTexture);
this.Controls.Add(this.txtTexture);
```

```
            this.Controls.Add(this.label9);
            this.Controls.Add(this.btnCancel);
            this.Controls.Add(this.btnOK);
            this.Controls.Add(this.nmSpeed);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.grpMoon);
            this.Controls.Add(this.nmMass);
            this.Controls.Add(this.label1);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
            this.Name = "frmNewObjAdv";
            this.Text = "Add New Object";
            this.TopMost = true;
            this.Load += new System.EventHandler(this.frmNewObjAdv_Load);
            ((System.ComponentModel.ISupportInitialize)(this.nmMass)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmMoonMass)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmMoonRadius)).EndInit();
            this.grpMoon.ResumeLayout(false);
            this.grpMoon.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.nmSpeed)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmRotation)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.NumericUpDown nmMass;
        private System.Windows.Forms.CheckBox ckAddMoon;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.NumericUpDown nmMoonMass;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.NumericUpDown nmMoonRadius;
        private System.Windows.Forms.GroupBox grpMoon;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.NumericUpDown nmSpeed;
        private System.Windows.Forms.Button btnOK;
        private System.Windows.Forms.Button btnCancel;
        private System.Windows.Forms.Label lbRadius;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.TextBox txtTexture;
        private System.Windows.Forms.Button btnLoadTexture;
        private System.Windows.Forms.OpenFileDialog openFileDialog1;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.NumericUpDown nmRotation;
        private System.Windows.Forms.CheckBox chFixed;
        private ctlDirection ctlDirection1;
    }
}
```

13 frmNewSim.cs

```
   /* AIMIS
Copyright (C) 2014, 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using OpenTK;

namespace AIMIS
{
    public partial class frmNewSim : Form
    {
        public gbVariables gbvars;
        public tkui MainUIclass;
        public System.Threading.Thread thMainUI;

        public frmNewSim()
        {
            InitializeComponent();
        }

        private void btnLaunch_Click(object sender, EventArgs e)
        {
            gbvars.NewObjectMass = 5f;
            gbvars.ShowTrails = true;
            gbvars.ShortTrails = true;
            Random rand = new Random();

            if(rbRandom.Checked)
            {
                //create random planets
                for (int ii = 0; ii < nmRandNumber.Value; ii++)
                {
                    tkui.PlanetObject p1 = new tkui.PlanetObject();
                    p1.Position = new Vector2(((float)rand.NextDouble() - 0.5f) * 8f * (
                        float)tbSpread.Value, ((float)rand.NextDouble() - 0.5f) * 8f * (
                        float)tbSpread.Value);
                    p1.Velocity = new Vector2(((float)rand.NextDouble() - 0.5f) * ((float)
                        tbSpeed.Value / 100), ((float)rand.NextDouble() - 0.5f) * ((float)
                        tbSpeed.Value / 100));
                    p1.Mass = (float)rand.NextDouble() + (float)tbMass.Value / 20f;
                    p1.Trails = new List<Vector2>();
                    MainUIclass.lstPlanets.Add(p1);
                }
            }

            if (rbSimple.Checked)
            {

                MainUIclass.NewPlanet(1000f, 0f, 0f, 4.47E-5f, 0f);
                MainUIclass.NewPlanet(3f, 0f, -3f, -0.0149f, 0f);

            }

            if (rbEarthOrbit.Checked)
            {
                //lauch the earth orbit simulator
                frmEarthOrbit formearth = new frmEarthOrbit();
                formearth.MainUIclass = MainUIclass;
```

```
                formearth.thMainUI = thMainUI;
                formearth.gbvars = gbvars;
                formearth.Show();
                this.Hide();

                MainUIclass.NewPlanet(10000f, 0f, 0f, 0f, 0f, "earth.png", -0.002f, true);
                MainUIclass.blGeoStat = true;

            }

            //launch the tkui thread
            thMainUI.Start();


            this.Close();
        }

        private void btnOpenSaved_Click(object sender, EventArgs e)
        {
            openFileDialog1.ShowDialog();
        }

        private void openFileDialog1_FileOk(object sender, CancelEventArgs e)
        {
            gbvars.NewObjectMass = 5f;
            gbvars.ShowTrails = true;
            gbvars.ShortTrails = true;

            MainUIclass.LoadPlanets(openFileDialog1.FileName);

            thMainUI.Start();

            this.Close();
        }

        private void rbRandom_CheckedChanged(object sender, EventArgs e)
        {
            grbRand.Visible = rbRandom.Checked;
        }
    }
}
```

14 frmNewSim.Designer.cs

```
namespace AIMIS
{
    partial class frmNewSim
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
            otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.rbEmpty = new System.Windows.Forms.RadioButton();
            this.rbRandom = new System.Windows.Forms.RadioButton();
            this.rbSimple = new System.Windows.Forms.RadioButton();
            this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
            this.grbRand = new System.Windows.Forms.GroupBox();
            this.tbSpread = new System.Windows.Forms.TrackBar();
            this.label4 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.nmRandNumber = new System.Windows.Forms.NumericUpDown();
            this.label1 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.tbSpeed = new System.Windows.Forms.TrackBar();
            this.tbMass = new System.Windows.Forms.TrackBar();
            this.rbEarthOrbit = new System.Windows.Forms.RadioButton();
            this.btnOpenSaved = new System.Windows.Forms.Button();
            this.btnLaunch = new System.Windows.Forms.Button();
            this.grbRand.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.tbSpread)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmRandNumber)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.tbSpeed)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.tbMass)).BeginInit();
            this.SuspendLayout();
            //
            // rbEmpty
            //
            this.rbEmpty.AutoSize = true;
            this.rbEmpty.Location = new System.Drawing.Point(12, 12);
            this.rbEmpty.Name = "rbEmpty";
            this.rbEmpty.Size = new System.Drawing.Size(105, 17);
            this.rbEmpty.TabIndex = 0;
            this.rbEmpty.TabStop = true;
            this.rbEmpty.Text = "Empty␣Simulation";
            this.rbEmpty.UseVisualStyleBackColor = true;
            //
            // rbRandom
            //
            this.rbRandom.AutoSize = true;
            this.rbRandom.Location = new System.Drawing.Point(12, 35);
            this.rbRandom.Name = "rbRandom";
            this.rbRandom.Size = new System.Drawing.Size(90, 17);
```

```
this.rbRandom.TabIndex = 1;
this.rbRandom.TabStop = true;
this.rbRandom.Text = "Random Start";
this.rbRandom.UseVisualStyleBackColor = true;
this.rbRandom.CheckedChanged += new System.EventHandler(this.
    rbRandom_CheckedChanged);
//
// rbSimple
//
this.rbSimple.AutoSize = true;
this.rbSimple.Location = new System.Drawing.Point(12, 58);
this.rbSimple.Name = "rbSimple";
this.rbSimple.Size = new System.Drawing.Size(81, 17);
this.rbSimple.TabIndex = 2;
this.rbSimple.TabStop = true;
this.rbSimple.Text = "Simple Orbit";
this.rbSimple.UseVisualStyleBackColor = true;
//
// openFileDialog1
//
this.openFileDialog1.FileName = "openFileDialog1";
this.openFileDialog1.Filter = "XML files|*.xml|All files|*.*";
this.openFileDialog1.FileOk += new System.ComponentModel.CancelEventHandler(
    this.openFileDialog1_FileOk);
//
// grbRand
//
this.grbRand.Controls.Add(this.tbSpread);
this.grbRand.Controls.Add(this.label4);
this.grbRand.Controls.Add(this.label2);
this.grbRand.Controls.Add(this.nmRandNumber);
this.grbRand.Controls.Add(this.label1);
this.grbRand.Controls.Add(this.label3);
this.grbRand.Controls.Add(this.tbSpeed);
this.grbRand.Controls.Add(this.tbMass);
this.grbRand.Location = new System.Drawing.Point(123, 12);
this.grbRand.Name = "grbRand";
this.grbRand.Size = new System.Drawing.Size(220, 202);
this.grbRand.TabIndex = 5;
this.grbRand.TabStop = false;
this.grbRand.Text = "Random Start";
this.grbRand.Visible = false;
//
// tbSpread
//
this.tbSpread.Location = new System.Drawing.Point(9, 156);
this.tbSpread.Minimum = 1;
this.tbSpread.Name = "tbSpread";
this.tbSpread.Size = new System.Drawing.Size(205, 45);
this.tbSpread.TabIndex = 7;
this.tbSpread.Value = 1;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(20, 140);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(41, 13);
this.label4.TabIndex = 6;
this.label4.Text = "Spread";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(20, 41);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(79, 13);
this.label2.TabIndex = 4;
this.label2.Text = "Average speed";
//
// nmRandNumber
//
this.nmRandNumber.Location = new System.Drawing.Point(105, 14);
```

```
this.nmRandNumber.Maximum = new decimal(new int[] {
10000,
0,
0,
0});
this.nmRandNumber.Name = "nmRandNumber";
this.nmRandNumber.Size = new System.Drawing.Size(109, 20);
this.nmRandNumber.TabIndex = 1;
this.nmRandNumber.Value = new decimal(new int[] {
500,
0,
0,
0});
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(6, 16);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(93, 13);
this.label1.TabIndex = 0;
this.label1.Text = "Number␣of␣objects";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(20, 89);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(74, 13);
this.label3.TabIndex = 5;
this.label3.Text = "Average␣mass";
//
// tbSpeed
//
this.tbSpeed.Location = new System.Drawing.Point(9, 57);
this.tbSpeed.Maximum = 100;
this.tbSpeed.Name = "tbSpeed";
this.tbSpeed.Size = new System.Drawing.Size(205, 45);
this.tbSpeed.TabIndex = 2;
this.tbSpeed.TickFrequency = 10;
//
// tbMass
//
this.tbMass.Location = new System.Drawing.Point(9, 108);
this.tbMass.Maximum = 100;
this.tbMass.Name = "tbMass";
this.tbMass.Size = new System.Drawing.Size(205, 45);
this.tbMass.TabIndex = 3;
this.tbMass.TickFrequency = 10;
//
// rbEarthOrbit
//
this.rbEarthOrbit.AutoSize = true;
this.rbEarthOrbit.Location = new System.Drawing.Point(12, 81);
this.rbEarthOrbit.Name = "rbEarthOrbit";
this.rbEarthOrbit.Size = new System.Drawing.Size(75, 17);
this.rbEarthOrbit.TabIndex = 6;
this.rbEarthOrbit.TabStop = true;
this.rbEarthOrbit.Text = "Earth␣Orbit";
this.rbEarthOrbit.UseVisualStyleBackColor = true;
//
// btnOpenSaved
//
this.btnOpenSaved.Image = global::AIMIS.Properties.Resources.document_open;
this.btnOpenSaved.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
this.btnOpenSaved.Location = new System.Drawing.Point(12, 144);
this.btnOpenSaved.Name = "btnOpenSaved";
this.btnOpenSaved.Size = new System.Drawing.Size(105, 32);
this.btnOpenSaved.TabIndex = 4;
this.btnOpenSaved.Text = "Load␣saved␣";
this.btnOpenSaved.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
this.btnOpenSaved.UseVisualStyleBackColor = true;
this.btnOpenSaved.Click += new System.EventHandler(this.btnOpenSaved_Click);
```

```
            //
            // btnLaunch
            //
            this.btnLaunch.Image = global::AIMIS.Properties.Resources.fork;
            this.btnLaunch.ImageAlign = System.Drawing.ContentAlignment.MiddleRight;
            this.btnLaunch.Location = new System.Drawing.Point(12, 106);
            this.btnLaunch.Name = "btnLaunch";
            this.btnLaunch.Size = new System.Drawing.Size(105, 32);
            this.btnLaunch.TabIndex = 3;
            this.btnLaunch.Text = "Launch";
            this.btnLaunch.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            this.btnLaunch.UseVisualStyleBackColor = true;
            this.btnLaunch.Click += new System.EventHandler(this.btnLaunch_Click);
            //
            // frmNewSim
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(355, 226);
            this.Controls.Add(this.rbEarthOrbit);
            this.Controls.Add(this.grbRand);
            this.Controls.Add(this.btnOpenSaved);
            this.Controls.Add(this.btnLaunch);
            this.Controls.Add(this.rbSimple);
            this.Controls.Add(this.rbRandom);
            this.Controls.Add(this.rbEmpty);
            this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow;
            this.Name = "frmNewSim";
            this.Text = "New Simulation";
            this.TopMost = true;
            this.grbRand.ResumeLayout(false);
            this.grbRand.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.tbSpread)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.nmRandNumber)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.tbSpeed)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.tbMass)).EndInit();
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.RadioButton rbEmpty;
        private System.Windows.Forms.RadioButton rbRandom;
        private System.Windows.Forms.RadioButton rbSimple;
        private System.Windows.Forms.Button btnLaunch;
        private System.Windows.Forms.Button btnOpenSaved;
        private System.Windows.Forms.OpenFileDialog openFileDialog1;
        private System.Windows.Forms.GroupBox grbRand;
        private System.Windows.Forms.NumericUpDown nmRandNumber;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TrackBar tbMass;
        private System.Windows.Forms.TrackBar tbSpeed;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TrackBar tbSpread;
        private System.Windows.Forms.RadioButton rbEarthOrbit;
    }
}
```

15 gbVariables.cs

```
   /* AIMIS
Copyright (C) 2014, 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AIMIS
{
    public class gbVariables
    {
        //show trails?
        public bool ShowTrails;
        //use short trails - helps performance
        public bool ShortTrails;
        //mass of objects added
        public float NewObjectMass;
        //for graph
        public List<float> lstVelocities = new List<float>();
        public bool blFollowObject = false;
        public int intDispObToFollow = 0;
        public int intObjectToTrack = 0;
        public bool blGraphTrack = false;
        public bool blTrackNewObject = true;


        //gravitational constant
        public float G = 0.0000006673f;

        //add moon with new object
        public bool blAddMoon = false;

        //add new object advanced option
        public bool blAddObjAdvanced = false;



    }
}
```

16 Program.cs

```
/* AIMIS
Copyright (C) 2014, 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Drawing;
using System.Collections.Generic;
using OpenTK;
using OpenTK.Graphics;
using OpenTK.Graphics.OpenGL;
using OpenTK.Input;
using System.Windows.Forms;

namespace AIMIS
{
    class MyApplication
    {
        [STAThread]
        public static void Main()
        {

            //load the main form, and instantiate gbVariables
            frmControl form = new frmControl();
            gbVariables gbvars = new gbVariables();
            form.gbvars = gbvars;

            //lauch the application
            Application.EnableVisualStyles();
            Application.Run(form);

        }
    }
}
```

17 tkui.cs

```
    /* AIMIS
Copyright (C) 2014, 2015 Alexis Enston
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details. */
using System;
using System.Drawing;
using System.Collections.Generic;
using OpenTK;
using OpenTK.Graphics;
using OpenTK.Graphics.OpenGL;
using OpenTK.Input;
using System.IO;
using System.Xml.Serialization;
using System.Drawing.Imaging;

namespace AIMIS
{
        public class tkui
        {
        public gbVariables gbvars;

        //serialize the lstPlanets to the filename given
        public void SavePlanets(string filename)
        {
            XmlSerializer xmlser = new XmlSerializer(typeof (List<PlanetObject>));

            using(Stream stream = File.Create(filename)) {
                    xmlser.Serialize(stream, lstPlanets);

            }

        }

        //unserialize lstPlanets from the file
        public void LoadPlanets(string filename)
        {
           XmlSerializer xmlser = new XmlSerializer(typeof (List<PlanetObject>));

           using (Stream stream = File.Open(filename, FileMode.Open))
           {
               lstPlanets = xmlser.Deserialize(stream) as List<PlanetObject>;
           }

            //reset textures
           foreach (PlanetObject planob in lstPlanets)
           {
               planob.Texture = 0;
           }


        }

        //Add an object / planet to the simulation
        public void NewPlanet(float Mass, float PosX, float PosY, float VelX, float VelY,
            string TextureFilename = null, float Rotation = 0, bool Fixed = false)
        {
            tkui.PlanetObject p2 = new tkui.PlanetObject();
            p2.Mass = Mass;
            p2.Position = new Vector2(PosX, PosY);
            p2.Velocity = new Vector2(VelX, VelY);
            p2.Trails = new List<Vector2>();
            p2.BitmapFilename = TextureFilename;
            p2.RotationAngle = 0f;
            p2.RotationTime = Rotation;
            p2.Fixed = Fixed;
```

```
        lstPlanets.Add(p2);
    }

    //draw a circle on the opentk gamewindow
            public void DrawCircle (int segments, float xpos, float ypos, float radius
                )
            {

                    GL.Begin (PrimitiveType.Polygon);
                    for (int i = 0; i < segments; i ++) {
                            float theta = (2.0f * (float)Math.PI * i) / (float)
                                segments;
                            float cxx = radius * (float)Math.Cos (theta);
                            float cyy = radius * (float)Math.Sin (theta);
                            GL.Vertex2 (xpos + cxx, ypos + cyy);
                    }
                    GL.End ();
            }


    //the class for each object
            public class PlanetObject
            {

                    private float mass;
        private float radius;
        //mass
                    public float Mass {
                            get {
                                    return this.mass;
                            }
                            set {
                                    this.mass = value;
                this.radius = (float)Math.Pow ((value * 3) / (Math.PI * 4 * 8000), (
                    double)1 / 3);
                            }
                    }
        //radius calculated from mass
                    public float Radius {
                            get {
                                    return this.radius;
                            }
            set
            {
                this.radius = value;
            }
                    }

        //position and movement
                    public Vector2 Velocity;
                    public Vector2 Position;
        //Trails
                    public List<Vector2> Trails;
        //Rotation - angle to add each step
        //Used for planets with textures
        public float RotationTime;
        public float RotationAngle;
        //texture
        private int texture;
        //load texture if called
        public int Texture
        {
            get
            {
                if (this.texture > 0)
                    return this.texture;
                else if (this.BitmapFilename != null)
                {
                    this.texture = tkui.LoadTexture(new Bitmap(this.BitmapFilename));
                    return this.texture;
                }
                else
                    return 0;
```

```
            }
            set
            {
                this.texture = value;
            }
        }

        //public Bitmap BitmapTexture;
        public string BitmapFilename;
        //dont move this object, ie keep it at center for earth sims
        public bool Fixed;
            }

    public void ClearTrails()
    {
        //delete trails
        lstTrails = new List<List<Vector2>>();

        foreach (PlanetObject planobj in lstPlanets)
        {
            planobj.Trails = new List<Vector2>();
        }
    }

    //get the position of the mouse as a vector scaled to the gamewindow
    public Vector2 MousePosition(float mX, float mY, GameWindow game)
    {
        Vector2 vecMousePos = new Vector2(((mX) / (float)game.Width - 0.5f) * game.
            Width * 2 *
            ZoomMulti - ViewPointV.X * game.Width * ZoomMulti, 0 - ((mY) / (float)game
                .Height - 0.5f)
            * game.Height * 2 * ZoomMulti - ViewPointV.Y * game.Height * ZoomMulti);
        return vecMousePos;

    }

            //For the textures - ie image of earth
    //loads a bitmap, and returns it's location as an int
            public static int LoadTexture(Bitmap bitmap)
            {
                    int texture;
                    GL.GenTextures(1, out texture);
                    GL.BindTexture(TextureTarget.Texture2D, texture);

                    BitmapData data = bitmap.LockBits(new Rectangle(0, 0, bitmap.Width
                        , bitmap.Height),
                                                      ImageLockMode.ReadOnly, System.
                                                          Drawing.Imaging.PixelFormat.
                                                          Format32bppArgb);

                    GL.TexImage2D(TextureTarget.Texture2D, 0, PixelInternalFormat.Rgba
                        , data.Width, data.Height, 0,
                                  OpenTK.Graphics.OpenGL.PixelFormat.Bgra, PixelType.
                                      UnsignedByte, data.Scan0);

                    bitmap.UnlockBits(data);
                    bitmap.Dispose();


                    GL.TexParameter(TextureTarget.Texture2D, TextureParameterName.
                        TextureMinFilter, (int)All.Linear);


                    return texture;
            }

    //variables
    //list of planets
    public List<PlanetObject> lstPlanets = new List<PlanetObject>();
    //list of trails of 'dead' planets
    List<List<Vector2>> lstTrails = new List<List<Vector2>>();
    //color of planets
    Color colPlanets = Color.LightYellow;
```

```
//viewpoint
Vector3 ViewPointV = new Vector3(0f, 0f, 0f);
public float ZoomMulti = 0.01f;
//speed
public int SimulationSpeed = 20;
//Draw lines for showing geostyationary orbit?
public bool blGeoStat = false;
//Show dot on earth for geostat
public bool blShowGeostatDot = false;
public float fAngleGeostat = 0f;

[STAThread]
        public void Main ()
        {

                using (var game = new GameWindow(700,500, new GraphicsMode
                    (32,24,0,4))) {

        //run at 60fps
        game.TargetRenderFrequency = 60;

                        Matrix4 matrix = Matrix4.CreateTranslation (0, 0, 0);
                        game.Load += (sender, e) =>
                        {
                                // setup settings, load textures, sounds
                                game.VSync = VSyncMode.On;
            game.Title = "AIMIS Simulation";
                        };

                        game.Resize += (sender, e) =>
                        {
                                GL.Viewport (0, 0, game.Width, game.Height);
                        };

        //mouse click to add logic
        Vector2 MoCinitialvec = new Vector2(0f, 0f);
        Vector2 MoCdvec = new Vector2(0f, 0f);
        bool MoCdraw = false;
        Vector3 PrevViewpoint = new Vector3(0f, 0f, 0f);

                        //enable textures
                        GL.Enable (EnableCap.Texture2D);
                        GL.BlendFunc(BlendingFactorSrc.SrcAlpha,
                            BlendingFactorDest.OneMinusSrcAlpha);


        //mouse click event
        game.Mouse.ButtonDown += (sender, e) =>
            {
                MoCinitialvec = MousePosition(e.X, e.Y, game);
                MoCdvec = MoCinitialvec;

                //start adding an object
                if (e.Button == MouseButton.Left)
                {
                    MoCdraw = true;
                }

                //start moving viewpoint
                if (e.Button == MouseButton.Right) {
                    PrevViewpoint = ViewPointV;
                }


            };

        game.Mouse.ButtonUp += (sender, e) =>
            {
                //if left click - add object
                if (e.Button == MouseButton.Left)
                {
                    //show advanced options?
                    if (gbvars.blAddObjAdvanced)
```

```
{
    frmNewObjAdv frmobj = new frmNewObjAdv();
    frmobj.gbvars = gbvars;
    frmobj.ShowDialog();
    if (frmobj.DialogResult == System.Windows.Forms.
        DialogResult.OK)
    {

        PlanetObject plan = new PlanetObject();
        plan.Mass = frmobj.fMass;
        plan.Position = MoCinitialvec;
        plan.Velocity = new Vector2((float)Math.Cos(frmobj.
            fAngle) * frmobj.fSpeed, -(float)Math.Sin(frmobj.
            fAngle) * frmobj.fSpeed);
        plan.Trails = new List<Vector2>();
        if (File.Exists(frmobj.stTextureFilename))
        {
            plan.BitmapFilename = frmobj.stTextureFilename;
        }
        plan.RotationAngle = 0f;
        plan.RotationTime = frmobj.fRotation;
        plan.Fixed = frmobj.blFixed;
        lstPlanets.Add(plan);

        //are we graphing the speed of the new planets?
        if (gbvars.blTrackNewObject)
        {
            gbvars.intObjectToTrack = lstPlanets.Count - 1;
            gbvars.lstVelocities.Clear();
        }

        //add a moon?
        if (frmobj.blAddMoon)
        {
            float distance = frmobj.fMoonDistance;
            PlanetObject moon = new PlanetObject();
            moon.Mass = frmobj.fMoonMass;
            moon.Position = MoCinitialvec;
            moon.Position.X += distance;
            moon.Velocity = plan.Velocity; //(MoCdvec -
                MoCinitialvec) * 0.05f;
            moon.Velocity.Y += (float)Math.Sqrt((gbvars.G * (
                moon.Mass + plan.Mass)) / distance);
            moon.Trails = new List<Vector2>();
            lstPlanets.Add(moon);
        }
    }
}

//otherwise add object without any dialogue
else
{

    MoCdvec = MousePosition(e.X, e.Y, game);
    PlanetObject plan = new PlanetObject();
    plan.Mass = gbvars.NewObjectMass;
    plan.Position = MoCinitialvec;
    plan.Velocity = (MoCdvec - MoCinitialvec) * 0.05f;
    plan.Trails = new List<Vector2>();
    lstPlanets.Add(plan);

    //are we graphing the speed of the new planets?
    if (gbvars.blTrackNewObject)
    {
        gbvars.intObjectToTrack = lstPlanets.Count - 1;
        gbvars.lstVelocities.Clear();
    }

    //add a moon?
    if (gbvars.blAddMoon)
    {
        float distance = plan.Radius * 3;
        PlanetObject moon = new PlanetObject();
```

```
                    moon.Mass = gbvars.NewObjectMass / 10;
                    moon.Position = MoCinitialvec;
                    moon.Position.X += distance;
                    moon.Velocity = (MoCdvec - MoCinitialvec) * 0.05f;
                    moon.Velocity.Y += (float)Math.Sqrt((gbvars.G * (moon.
                        Mass + plan.Mass)) / distance);
                    moon.Trails = new List<Vector2>();
                    lstPlanets.Add(moon);
                }
            }

            MoCdraw = false;
        }
    };

//scroll wheel - zoom in / out
game.Mouse.WheelChanged += (sender, e) =>
    {

        if (ZoomMulti - e.DeltaPrecise * 0.001f > 0)
        {
            ZoomMulti -= e.DeltaPrecise * 0.001f;
        }

    };

//moving mouse
game.Mouse.Move += (sender, e) =>
    {
        //for adding object
        MoCdvec = MousePosition(e.X, e.Y, game);

        //for moving viewpoint
        if(e.Mouse.RightButton == ButtonState.Pressed)
        {
            ViewPointV = new Vector3((MoCdvec - MoCinitialvec).X / (game.
                Width * ZoomMulti), (MoCdvec - MoCinitialvec).Y / (game.
                Height * ZoomMulti), 0) + PrevViewpoint;
        }

    };

//keyboard input
game.KeyPress += (sender, e) =>
    {
        switch (e.KeyChar)
        {
            case 'f':
                game.WindowState = WindowState.Fullscreen;
                break;
            case 'c':
                gbvars.ShowTrails = !gbvars.ShowTrails;
                break;
            case 't':
                lstTrails = new List<List<Vector2>> ();
                                    foreach (PlanetObject planobj in
                                        lstPlanets) {
                                            planobj.Trails = new List<
                                                Vector2> ();
                                    }
                break;
        }

    };

//more keyboard input
                game.UpdateFrame += (sender, e) =>
                {
                        if (game.Keyboard [Key.Escape]) {
                                game.Exit ();
                        }
                        if (game.Keyboard [Key.A]) {
                                ViewPointV.X += 0.01f;
```

```
                        }
                        if (game.Keyboard [Key.D]) {
                                ViewPointV.X -= 0.01f;
                        }
                        if (game.Keyboard [Key.W]) {
                                ViewPointV.Y -= 0.01f;
                        }
                        if (game.Keyboard [Key.S]) {
                                ViewPointV.Y += 0.01f;
                        }
                        if (game.Keyboard [Key.Z]) {
                                ZoomMulti += 0.001f;
                        }
                        if (game.Keyboard [Key.X] && ZoomMulti > 0.001f) {
            ZoomMulti -= 0.001f;
                        }

                };

        //for slowing down simulation
        int SimulationSlowDownStep = 0;

                game.RenderFrame += (sender, e) =>
                {
                        // render graphics
            //clears screen
                        GL.Clear (ClearBufferMask.ColorBufferBit |
                            ClearBufferMask.DepthBufferBit);

                        GL.MatrixMode (MatrixMode.Projection);

            Vector2 followPosition = new Vector2(0, 0);

                        //trackobject
            if (gbvars.blFollowObject && lstPlanets.Count > gbvars.
                intDispObToFollow)
            {
                followPosition = lstPlanets[gbvars.intDispObToFollow].Position;
                ViewPointV.X = -followPosition.X / (game.Width * ZoomMulti);
                ViewPointV.Y = -followPosition.Y / (game.Height * ZoomMulti);
            }

            matrix = Matrix4.CreateTranslation(ViewPointV);

                        GL.LoadMatrix (ref matrix);
                        GL.Ortho (-game.Width * ZoomMulti, game.Width *
                            ZoomMulti, -game.Height * ZoomMulti, game.
                            Height * ZoomMulti, 0.0, 4.0);

                        //speedup / slowdown
            //slowdown?
            if (SimulationSpeed < 20)
            {
                //Stop simulation when sim speed = 0
                if (SimulationSpeed == 0)
                    SimulationSlowDownStep = 1;

                SimulationSlowDownStep += SimulationSpeed;
                if (SimulationSlowDownStep > 20)
                    SimulationSlowDownStep = 0;
            }
            else
                SimulationSlowDownStep = 0;


                        for(int zx = 20; (zx < SimulationSpeed || zx ==
                            20) && SimulationSlowDownStep == 0; zx ++) {

                //keep list of speeds for graphs
                if (lstPlanets.Count > gbvars.intObjectToTrack)
                {
                    gbvars.lstVelocities.Add(lstPlanets[gbvars.intObjectToTrack].
                        Velocity.Length);
```

```
            if (lstPlanets.Count > 500)
            {
                lstPlanets.RemoveAt(0);
            }
        }

                    //calculate forces between objects
                    for (int i = lstPlanets.Count - 1; i >= 0; i--) {

                            PlanetObject planob = lstPlanets [i];

                            for (int ic = lstPlanets.Count - 1; ic >=
                                0; ic--) {
                                PlanetObject plan2 = lstPlanets [
                                    ic];
                                if (plan2.Position != planob.
                                    Position) {

                                        //collision detection,
                                            merge objects
                    //check if they overlap, and if we have a fixed object
                    if ((planob.Position - plan2.Position).Length < planob.
                        Radius)
                    {
                        //if fixed, keep it
                        if (plan2.Fixed)
                        {
                            planob = plan2;
                        }
                        if (planob.Fixed != true)
                        {
                            Vector2 CombVelocity = planob.Velocity * planob.
                                Mass + plan2.Velocity * plan2.Mass;
                            //Keep texture of largest object
                            if (planob.Mass < plan2.Mass)
                                planob.Texture = plan2.Texture;

                            planob.Mass += plan2.Mass;
                            planob.Velocity = CombVelocity / planob.Mass;
                        }
                        //add 'dead' objects trails to the other list
                        lstTrails.Add(plan2.Trails);

                        //Delete the 'dead' object
                        lstPlanets.RemoveAt(ic);
                        i--;
                    }

                    else
                    {
                        //calculate force with vectors
                        //Skip this if we have a 'fixed' object
                        if (planob.Fixed != true)
                        {
                            //distance squared
                            float dissqu = (planob.Position - plan2.Position).
                                Length;
                            dissqu = dissqu * dissqu;
                            Vector2 Force = -gbvars.G * ((planob.Mass * plan2.
                                Mass) / dissqu) * ((planob.Position - plan2.
                                Position) / (float)Math.Sqrt(dissqu));
                            Vector2 Acceleration = Force / planob.Mass;
                            planob.Velocity += Acceleration;
                        }
                    }
                                        }
                                }
                            }


                    foreach (PlanetObject planob in lstPlanets) {
                //add a 'step' to the planet
                planob.Position += planob.Velocity;
```

```
            //add the trails
            //are we doing short trails?
                                    if(gbvars.ShortTrails && planob.Trails
                                        .Count > 500 ) {
                                            planob.Trails.RemoveAt(0);
                                    }

            planob.Trails.Add(planob.Position);
                                }
                        }

    //Now draw the objects to the gamewindow

    GL.Color3(Color.DarkRed);

    //draw vector [dead] trails
    if (gbvars.ShowTrails)
    {
        foreach (List<Vector2> TrailL in lstTrails)
        {
            GL.Begin(PrimitiveType.LineStrip);
            foreach (Vector2 pos in TrailL)
            {
                GL.Vertex2(pos.X, pos.Y);
            }
            GL.End();
        }
    }

                        //draw planets
                        for (int i = lstPlanets.Count - 1; i >= 0; i--) {
        if (i == gbvars.intObjectToTrack && gbvars.blGraphTrack)
            GL.Color3(Color.Yellow);
        else
        {
            if (gbvars.blFollowObject && i == gbvars.intDispObToFollow)
                GL.Color3(Color.Blue);
            else
                GL.Color3(colPlanets);
        }

                                PlanetObject planob = lstPlanets [i];

        //spin a planet, but only do it if we have a texture, otherwise
            its pointless
        //also only do it if we're doing the 'step'
        if (planob.Texture > 0 && SimulationSlowDownStep == 0)
        {
            if (SimulationSpeed < 20)
                planob.RotationAngle += planob.RotationTime;
            else
                planob.RotationAngle += planob.RotationTime * (
                    SimulationSpeed - 19);
            if (planob.RotationAngle > (float)Math.PI * 2)
                planob.RotationAngle -= (float)Math.PI * 2;
            if (planob.RotationAngle < -(float)Math.PI * 2)
                planob.RotationAngle += (float)Math.PI * 2;
        }

        if (planob.Texture > 0)
        {
            //Draw planet with texture
            GL.BindTexture(TextureTarget.Texture2D, planob.Texture);
            GL.Enable(EnableCap.Blend);

            //don't recolor the texture
            GL.Color3(Color.White);

            //rotate?
            Matrix4 rotmatrix = Matrix4.CreateFromAxisAngle(new Vector3(0,
                 0, 1), planob.RotationAngle);
            rotmatrix = rotmatrix * Matrix4.CreateTranslation(planob.
```

```
                Position.X, planob.Position.Y, 0);

            GL.MultMatrix(ref rotmatrix);

            GL.MatrixMode(MatrixMode.Modelview);

            //draw square for mapping of texture
            GL.Begin(PrimitiveType.Quads);
            GL.TexCoord2(0, 0);
            GL.Vertex2(- planob.Radius,  - planob.Radius);
            GL.TexCoord2(1, 0);
            GL.Vertex2(planob.Radius, - planob.Radius);
            GL.TexCoord2(1, 1);
            GL.Vertex2( planob.Radius, planob.Radius);
            GL.TexCoord2(0, 1);
            GL.Vertex2(- planob.Radius, planob.Radius);
            GL.End();

            //map texture
            GL.BindTexture(TextureTarget.Texture2D, 0);


            GL.MatrixMode(MatrixMode.Projection);
            GL.LoadMatrix(ref matrix);
            GL.Ortho(-game.Width * ZoomMulti, game.Width * ZoomMulti, -
                game.Height * ZoomMulti, game.Height * ZoomMulti, 0.0,
                4.0);


        }

        else
        {
            //draw a circle to represent object
            DrawCircle(30, planob.Position.X, planob.Position.Y, planob.
                Radius);

        }

        GL.Color3(Color.DarkRed);


                                //draw trails
        if (gbvars.ShowTrails)
        {

                                        GL.Begin (PrimitiveType.LineStrip)
                                            ;

            for (int j = 0; j < planob.Trails.Count; j++)
            {
                GL.Vertex2(planob.Trails[j].X, planob.Trails[j].Y);
            }
                                        GL.End ();
                                }

                        }

    //draw mouse click line, for when adding object
    if (MoCdraw)
    {
        GL.Begin(PrimitiveType.Lines);
        GL.Color3(Color.Yellow);
        GL.Vertex2(MoCinitialvec);
        GL.Vertex2(MoCdvec);
        GL.End();

        //draw the object
        DrawCircle(30, MoCinitialvec.X, MoCinitialvec.Y, (float)Math.Pow((
            gbvars.NewObjectMass * 3) / (Math.PI * 4 * 8000), (double)1 /
            3));
    }
```

```
                    //Draw lines for geostationary simulation
                    if (blGeoStat && lstPlanets.Count > 1)
                    {
                        for (int i = 1; i < lstPlanets.Count; i++)
                        {
                            GL.Begin(PrimitiveType.Lines);
                            GL.Color3(Color.Aqua);
                            GL.Vertex2(lstPlanets[0].Position);
                            GL.Vertex2(lstPlanets[i].Position);
                            GL.End();
                        }
                    }

                    //Draw earth sim arrows
                    if (blShowGeostatDot)
                    {
                        GL.Color3(Color.Yellow);
                        DrawCircle(30, (float)Math.Sin(-lstPlanets[0].RotationAngle +
                            fAngleGeostat) * lstPlanets[0].Radius, (float)Math.Cos(-
                            lstPlanets[0].RotationAngle + fAngleGeostat) * lstPlanets[0].
                            Radius, 0.1f);
                    }

                    //load onto screen
                                    game.SwapBuffers ();
                                };

                //run the game loop
                game.Run();

                        }
                    }
                }
            }
```