```
using System;
using System.IO;
using System.Linq;
using System.Collections.Generic;
using Gtk;
using Cairo;
using static Program.Program;
using static Program.Constants;
using Structures;
namespace UI {
	public class Menu : Window {
		protected VBox containerbox;
		protected VBox radiobox;
		protected ScrolledWindow systemscrollbox;
		protected VBox systembox;
		protected HBox donebox;
		protected Scale TimestepScale;
		protected Scale RScale;
		protected Scale LineScale;
		protected RadioButton radio0;
		protected RadioButton radio1;
		protected RadioButton radio2;
		protected RadioButton radio3;
		protected RadioButton radio4;
		protected ComboBoxText bCombo;
		public Button loadButton {get; set;}
		protected Entry filename;
		protected readonly String SYSTEM_DIRECTORY = "ExampleSystems";
		protected static List<Structures.Body> std_bodies = Examples.solar_system_bodies;
		internal static List<BodyBox> new_bodies {get; set;} = new List<BodyBox>();
		public SaveData temp_savedata {get; set;} = null;
		protected static List<bool> centers = new List<bool>();

		public Menu(Gtk.WindowType s = Gtk.WindowType.Toplevel) : base(s) { // weird inheritancy stuff, don't change
			this.SetDefaultSize(300,400);
			this.DeleteEvent += delegate { Application.Quit (); };
			containerbox = new VBox(homogeneous: false, spacing: 3);
			radiobox = new VBox(homogeneous: false, spacing: 3);
			systemscrollbox = new ScrolledWindow();
			systembox = new VBox(homogeneous: false, spacing: 3);
			donebox = new HBox(homogeneous: false, spacing: 3);

			var l1 = new Label("Mechanics Timestep");
			var l2 = new Label("Planetary Radii Multiplier");
			var l3 = new Label("Orbit Trail Length");
			TimestepScale = new Scale(Orientation.Horizontal, 0.1,1000,0.1);
			TimestepScale.Value = 50;
			RScale = new Scale(Orientation.Horizontal, 1, 1000, 1);
			RScale.Value = 100;
			LineScale = new Scale(Orientation.Horizontal, 50, 1000, 1);
			LineScale.Value = 100;

			var addBox = new HBox();
			var addButton = new Button("Add");
			addButton.Clicked += new EventHandler(OnAddClick);
			filename = new Entry();
			var saveButton = new Button("Save");
			saveButton.Clicked += new EventHandler(OnSaveClick);
			loadButton = new Button("Load");
```

```
            loadButton.Clicked += new EventHandler(OnLoadClick);
            bCombo = new ComboBoxText();
            bCombo.AppendText("Custom");
            foreach (Body b in Menu.std_bodies) {
                bCombo.AppendText(b.name);
            }
            bCombo.Active = 0; // Default to Custom body
            addBox.PackStart(bCombo, true, false, 3);
            addBox.PackStart(addButton, true, false, 3);
            addBox.PackStart(filename, true, false, 3);
            addBox.PackStart(saveButton, true, false, 3);
            addBox.PackStart(loadButton, true, false, 3);
            var doneButton = new Button("Done");
            doneButton.Clicked += new EventHandler (OnDoneClick);
            var exitButton = new Button("Exit");
            exitButton.Clicked += new EventHandler(delegate{
                Application.Quit();
            });

            var optionsbox = new HBox(homogeneous: false, spacing: 3);
            var optionbox1 = new VBox(homogeneous: false, spacing: 3);
            var optionbox2 = new VBox(homogeneous: false, spacing: 3);
            var optionbox3 = new VBox(homogeneous: false, spacing: 3);
            optionbox1.PackStart(l1, true, true, 3);
            optionbox1.PackStart(TimestepScale, true, true, 3);
            optionbox2.PackStart(l2, true, true, 3);
            optionbox2.PackStart(RScale, true, true, 3);
            optionbox3.PackStart(l3, true, true, 3);
            optionbox3.PackStart(LineScale, true, true, 3);
            optionsbox.PackStart(optionbox1, true, true, 3);
            optionsbox.PackStart(optionbox2, true, true, 3);
            optionsbox.PackStart(optionbox3, true, true, 3);

            radiobox.PackStart(optionsbox, false, false, 3);
            radiobox.PackStart(addButton, false, false, 3);
            radiobox.PackStart(addBox, false, false, 3);
            systemscrollbox.Add(systembox);
            donebox.PackStart(doneButton, true, true, 3);
            donebox.PackStart(exitButton, true, true, 3);


            containerbox.PackStart(radiobox, false, false, 3);
            containerbox.PackStart(systemscrollbox, true, true, 3);
            containerbox.PackStart(donebox, false, false, 3);
            this.Add(containerbox);
            this.ShowAll();
        }
        protected void OnDoneClick(object obj, EventArgs args) {
            if (new_bodies.Count < 2) {
                Message("An empty system is not very interesting!");
                return;
            }
            try {
                Program.Program.CustomBodies.Clear();
                Program.Program.CustomCenters.Clear();
                centers.Clear();
                foreach (BodyBox b in new_bodies) {
                    b.Set();
                    Program.Program.CustomBodies.Add(b.body);
```

```
					centers.Add(b.CenterButton.Active);

				}
				Program.Program.CustomCenters = centers;
				Program.Program.radius_multiplier = RScale.Value;
				Program.Program.line_max = (int)LineScale.Value;
				Program.Program.timestep = TimestepScale.Value;
				Program.Program.Start();
				this.Destroy();
			} catch (Exception e) {
				Message("I'm sorry, something went wrong but I don't know what. \nIf you can find a bored developer, show him this stack trace:\n" + e.Message + e.StackTrace);
			}
		}
		protected void OnAddClick(object obj, EventArgs args) {

			var bodyBox = new BodyBox(menu: this, homogeneous: false, spacing: 3);
			String bString = bCombo.ActiveText;
			if (bString != "Custom") {
				var body = Examples.solar_system.First(b => b.name == bString);
				if (!(body.parent == null || new_bodies.Exists(b => b.name.Text == body.parent.name))) {
					body = std_bodies.First(b => b.name == bString);
				}
				bodyBox.body = body;
				bodyBox.ReverseSet();
			}

			bodyBox.name.Text = bCombo.ActiveText;
			systembox.PackStart(bodyBox, true, true, 3);
			new_bodies.Add(bodyBox);
			foreach (BodyBox b in new_bodies) {
				b.ResetParents();
			}
			this.ShowAll();
		}
		protected void OnSaveClick(object obj, EventArgs args) {
			if (filename.Text == "") {
				Message("Please enter a filename");
				return;
			}
			System.Xml.Serialization.XmlSerializer writer =
				new System.Xml.Serialization.XmlSerializer(typeof(SaveData));
			if (File.Exists(Environment.CurrentDirectory + "//" + filename.Text + ".xml")) {
				File.Delete(Environment.CurrentDirectory + "//" + filename.Text + ".xml");
			}
			FileStream file = File.Create(
				Environment.CurrentDirectory + "//" + filename.Text + ".xml");
			var bodies = new List<Body>();
			if (centers == null) centers = new List<bool>();
			centers.Clear();
			var elements = new List<OrbitalElements>();
			foreach (BodyBox b in new_bodies) {
				b.Set();
				bodies.Add(b.body);
				centers.Add(b.CenterButton.Active);
				elements.Add(new OrbitalElements() {
					semilatusrectum = b.SLRScale.Value*AU,
					eccentricity = b.EScale.Value,
					inclination = b.IncScale.Value*deg,
					ascendingNodeLongitude = b.ANLScale.Value*deg,
					periapsisArgument = b.PAScale.Value*deg,
					trueAnomaly = b.TAScale.Value*deg
				});
			}
			var data = new SaveData() {
				bodies = bodies,
				elements = elements,
				timestep = TimestepScale.Value,
				centers = centers,
				radius_multiplier = RScale.Value,
				line_max = LineScale.Value,
			};
			writer.Serialize(file, data);
			file.Close();
		}
		protected void OnLoadClick(object obj, EventArgs args) {
			System.Xml.Serialization.XmlSerializer reader =
				new System.Xml.Serialization.XmlSerializer(typeof(SaveData));
			SaveData data = new SaveData(); // To prevent compiler error
			if (temp_savedata != null) {
				data = temp_savedata;
				temp_savedata = null;
			} else {
				try {
					var file = new StreamReader(Environment.CurrentDirectory + "//" + filename.Text + ".xml");
					data = (SaveData)reader.Deserialize(file);
				} catch (IOException) {
					// Try in the system directory
					try {
						var file = new StreamReader(Environment.CurrentDirectory + "//" + SYSTEM_DIRECTORY + "//" + filename.Text + ".xml");
						data = (SaveData)reader.Deserialize(file);
					} catch (IOException) {
						Message("The specified file could not be found. Check that the name is spelt correctly and that it is in the correct directory");
						// cannot deserialize, exit
						return;
					}
				} catch (InvalidOperationException) {
					Message("The file is not a valid save file of this project");
					// cannot deserialize, exit
					return;
				}
			}
			RScale.Value = data.radius_multiplier;
			LineScale.Value = data.line_max;
			TimestepScale.Value = data.timestep;
			new_bodies.Clear();
			foreach (Widget w in systembox.Children) {
				if (w is BodyBox) systembox.Remove (w);
			}
			for (int i = 0; i < data.bodies.Count; i++) {
				var bbox = new BodyBox(menu: this, homogeneous: false, spacing: 3) {
					body = data.bodies[i],
				};
				bbox.CenterButton.Active = data.centers[i];
				if (data.elements != null && data.elements.Count != 0) {
					bbox.SetElements(data.elements[i]);
					bbox.ReverseSet(false);
				} else bbox.ReverseSet();
				new_bodies.Add(bbox);
				systembox.PackStart(bbox, true, true, 3);
			}
			foreach (BodyBox b in new_bodies) {
				b.ResetParents();
			}
			this.ShowAll();
		}
		protected void Message(String s) {
			var window = new Window("Message");
			var container = new VBox(homogeneous: true, spacing: 3);
			window.Add(container);
			container.PackStart(new Label(s), false, false, 3);
			var closeButton = new Button("Close");
			closeButton.Clicked += delegate {window.Destroy();};
			container.PackStart(closeButton, false, false, 3);
			window.ShowAll();
		}
		public void Remove(BodyBox b) {
			var name = b.name.Text;
			new_bodies.Remove(b);
			systembox.Remove(b);
			foreach (BodyBox a in new_bodies) {
				a.ResetParents();

			}
		}
	}
	public class BodyBox : HBox {
		public Body body {get; set;}
		public Entry name {get; set;}
		public ComboBoxText parent {get; set;} = new ComboBoxText();
		public Scale MassScale {get; set;}
		public Scale RadiusScale {get; set;}
		public Scale SLRScale {get; set;}
		public Scale EScale {get; set;}
		public Scale IncScale {get; set;}
		public Scale ANLScale {get; set;}
		public Scale PAScale {get; set;}
		public Scale TAScale {get; set;}
		public Scale RScale {get; set;}
		public Scale GScale {get; set;}
		public Scale BScale {get; set;}
		public CheckButton CenterButton {get; set;}
		public Button DeleteButton {get; set;}
		private static readonly double ECCENTRICITY_MAX = 3;
		private Menu menu;
		public BodyBox() {}
		public BodyBox(Menu menu, bool homogeneous = false, int spacing = 3) : base(homogeneous,spacing) {
			this.menu = menu;
			body = new Structures.Body();
			name = new Entry();
```

```
                        name.IsEditable = true;
                        ResetParents();
                        MassScale = new Scale(Orientation.Vertical, 0.1,50,0.01);
                        RadiusScale = new Scale(Orientation.Vertical, 0.1,1000000,0.1);
                        SLRScale = new Scale(Orientation.Vertical, 0.1,50,0.01);
                        EScale = new Scale(Orientation.Vertical, 0,ECCENTRICITY_MAX,0.001);
                        IncScale = new Scale(Orientation.Vertical, 0,180,0.01);
                        ANLScale = new Scale(Orientation.Vertical, 0,359.99,0.01);
                        PAScale = new Scale(Orientation.Vertical, 0,359.99,0.01);
                        TAScale = new Scale(Orientation.Vertical, 0,359.99,0.01);
                        RScale = new Scale(Orientation.Horizontal, 0, 1, 0.01);
                        GScale = new Scale(Orientation.Horizontal, 0, 1, 0.01);
                        BScale = new Scale(Orientation.Horizontal, 0, 1, 0.01);
                        CenterButton = new CheckButton("Focusable");
                        DeleteButton = new Button("Delete");
                        DeleteButton.Clicked += new EventHandler (OnDeleteClick);

                        parent.Changed += new EventHandler(OnParentChange);
                        MassScale.Inverted = true;
                        RadiusScale.Inverted = true;
                        SLRScale.Inverted = true;
                        EScale.Inverted = true;
                        IncScale.Inverted = true;
                        ANLScale.Inverted = true;
                        PAScale.Inverted = true;
                        TAScale.Inverted = true;
                        var mBox = new VBox(homogeneous: false, spacing: 3);
                        mBox.PackStart(new Label("ln(m)"), false, false, 3); mBox.PackStart(MassScale, true, true, 3);
                        var rBox = new VBox(homogeneous: false, spacing: 3);
                        rBox.PackStart(new Label("r (km)"), false, false, 3); rBox.PackStart(RadiusScale, true, true, 3);
                        var slrBox = new VBox(homogeneous: false, spacing: 3);
                        slrBox.PackStart(new Label("ρ (AU)"), false, false, 3); slrBox.PackStart(SLRScale, true, true, 3);
                        var eBox = new VBox(homogeneous: false, spacing: 3);
                        eBox.PackStart(new Label("e"), false, false, 3); eBox.PackStart(EScale, true, true, 3);
                        var incBox = new VBox(homogeneous: false, spacing: 3);
                        incBox.PackStart(new Label("i (°)"), false, false, 3); incBox.PackStart(IncScale, true, true, 3);
                        var anlBox = new VBox(homogeneous: false, spacing: 3);
                        anlBox.PackStart(new Label("Ω (°)"), false, false, 3); anlBox.PackStart(ANLScale, true, true, 3);
                        var paBox = new VBox(homogeneous: false, spacing: 3);
                        paBox.PackStart(new Label("ω (°)"), false, false, 3); paBox.PackStart(PAScale, true, true, 3);
                        var taBox = new VBox(homogeneous: false, spacing: 3);
                        taBox.PackStart(new Label("ν (°)"), false, false, 3); taBox.PackStart(TAScale, true, true, 3);

                        this.PackStart(name, true, true, 3);
                        this.PackStart(parent, false, false, 3);
                        this.PackStart(mBox, true, true, 3);
                        this.PackStart(rBox, true, true, 3);
                        this.PackStart(slrBox, true, true, 3);
                        this.PackStart(eBox, true, true, 3);
                        this.PackStart(incBox, true, true, 3);
                        this.PackStart(anlBox, true, true, 3);
                        this.PackStart(paBox, true, true, 3);
                        this.PackStart(taBox, true, true, 3);

                        var colorbox = new VBox(homogeneous: false, spacing: 3);
                        colorbox.PackStart(new Label("RGB"), false, false, 3);
                        colorbox.PackStart(RScale, true, true, 3);
                        colorbox.PackStart(GScale, true, true, 3);
                        colorbox.PackStart(BScale, true, true, 3);

                        this.PackStart(colorbox, true, true, 3);
                        var optionsbox = new VBox(homogeneous: false, spacing: 3);
                        optionsbox.PackStart(CenterButton, true, true, 3);
                        optionsbox.PackStart(DeleteButton, true, true, 3);
                        this.PackStart(optionsbox, true, true, 3);

                }
                protected void OnParentChange(object obj, EventArgs args) {
                        try {
                                var parentBody = Menu.new_bodies.FirstOrDefault(b => b.body.name == parent.ActiveText).body;
                                double hillrad = parentBody.HillRadius()/AU;
                                this.SLRScale.Digits = Math.Max(0,8);//3-(int)Math.Log(hillrad/100000));
                                this.SLRScale.SetIncrements(Math.Pow(10,-this.SLRScale.Digits),hillrad/100000);
                                this.SLRScale.SetRange(Math.Pow(10,-this.SLRScale.Digits),hillrad);
                        } catch (NullReferenceException) {} // no parent, don't set values
                }
                protected void OnDeleteClick(object obj, EventArgs args) {
            menu.Remove(this);
            menu.ShowAll();
                        this.Destroy();
        }
                public void Set() {
                        if (parent.ActiveText != this.name.Text && parent.Active != -1) {
                                var elements = new Structures.OrbitalElements() {
                                        semilatusrectum = SLRScale.Value*AU,
                                        eccentricity = EScale.Value,
                                        inclination = IncScale.Value*deg,
                                        ascendingNodeLongitude = ANLScale.Value*deg,
                                        periapsisArgument = PAScale.Value*deg,
                                        trueAnomaly = TAScale.Value*deg
                                };
                                body = new Structures.Body(Menu.new_bodies.FirstOrDefault(b => b.body.name == parent.ActiveText).body,elements);
                        }
                        body.name = this.name.Text;
                        body.stdGrav = Math.Pow(Math.E,MassScale.Value)*G*1e22;
                        body.radius = RadiusScale.Value*1e3;
                        body.color = new Vector3(RScale.Value, GScale.Value, BScale.Value);
                }
```

```
		public void SetElements(OrbitalElements elements) {
			try {
				if (elements.semilatusrectum > this.body.parent.HillRadius()/AU) {
					SLRScale.SetRange(1e-8,elements.semilatusrectum);
				}
			} catch (NullReferenceException) {} // body has no parent, we cannot check the slr
			SLRScale.Value = elements.semilatusrectum/AU;
			if (elements.eccentricity > ECCENTRICITY_MAX) {
				EScale.SetRange(0,elements.eccentricity);
			} else {
				EScale.SetRange(0, ECCENTRICITY_MAX); // there is no way to see the current range, so we'll set it every time
			}
			EScale.Value   = elements.eccentricity;
			IncScale.Value = elements.inclination/deg;
			ANLScale.Value = elements.ascendingNodeLongitude/deg;
			PAScale.Value  = elements.periapsisArgument/deg;
			TAScale.Value  = elements.trueAnomaly/deg;
		}
		public void ReverseSet(bool elem = true) {
			if (elem) try {
				parent.Active  = Menu.new_bodies.FindIndex(b => b.name.Text == body.parent.name);
				var elements   = new OrbitalElements(body.position-body.parent.position,body.velocity-body.parent.velocity,body.parent.stdGrav);
				this.SetElements(elements);
			} catch (NullReferenceException) {} // if body has no parent
			name.Text = body.name;
			MassScale.Value = Math.Log((body.stdGrav/G)/1e22);
			RadiusScale.Value = body.radius/1e3;
			RScale.Value = body.color.x;
			GScale.Value = body.color.y;
			BScale.Value = body.color.z;
		}
		public void ResetParents() {
			parent.RemoveAll();
			foreach (BodyBox b in Menu.new_bodies) {
				parent.AppendText(b.name.Text);
			}
			try {
				parent.Active = Menu.new_bodies.FindIndex(b => b.name.Text == body.parent.name);
			} catch (NullReferenceException) {} // parent no longer exists

		}
	}
	[Serializable()]
	public class SaveData {
		public List<Body> bodies {get; set;}
		public List<OrbitalElements> elements {get; set;}
		public List<bool> centers {get; set;}
		public double timestep {get; set;}
		public double radius_multiplier {get; set;}
		public double line_max {get; set;}
	}
}
```