SpaceGravity2D

1.4

Generated by Doxygen 1.8.12

Contents

Chapter 1

Introduction

Space gravity 2D makes much easier to build custom 3d solar system scenes with realistic gravitational motion. It can be used for implementing various gameplay types. Main features of this asset are: two different systems for simulating gravitational motion and tool for editing velocity vectors in unity scene window.

Two systems of gravitational simulation:

- Newtonian attraction calculation (N-body problem).
- Keplerian motion (2-body problem or 'RailMotion') is exactly predicted motion on static orbit, which behaviour is as if no outer gravitation exists. This method can be very useful for moving planets and moons whose orbits are not dynamically changing, as it has better performance and precision on long terms periods.

Step by step guide

- Attach SimulationControl component to any gameobject on the scene or just open SpaceGravity2d window, it will be created automatically.
- Attach CelestialBody component to gameobject.
- Configure body parameters, scale of attached sprite or mesh object, add colliders.
- Create second CelestialBody (it may be duplicate). Set mass lower than first body mass. Select motion type.
- Use 'find nearest attractor' button in second's body inspector or manually drag first body to Attractor property of second body to create planetary system. Button 'make circle orbit' may help to quick setup.
- Configure distance between bodies, their velocities with help of editor scene arrows, Gravitational parameters and TimeScale in SpaceGravity2D window.
- Add other bodies according to your wishes.

  Note that attractor field in CelestialBody component is required only for keplerian motion and for displaying orbits. If you prefer to not use keplerian motion at all, you don't need to set attractor for bodies.

Contacts

Email for support and suggestions: itanksp@gmail.com

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 SpaceGravity2D Namespace Reference

Orbital motion system.

Namespaces

- namespace Inspector

Classes

- class CelestialBody

  Component, which allows gameobject to attract other celestial bodies and be attracted by them.
- class CelestialBodySingle

  Component for standalone static orbiting body, which is independent from SimulationControl.
- class **CelestialBodyUtils**

  Math utility methods for help in orbits calculations.
- class OrbitData

  Orbit data container. Also contains methods for altering and updating orbit state.
- class OrbitDisplay

  Component for displaying current orbit of CelestialBody. CelestialBody should be attached to same Game Object.
- class PredictionSystem

  Basic prediction orbits calculator (singleton). Simulates whole scene with Euler n-body algorythm PointCount steps into future, and displays resulting orbits with linerenderers.
- class PredictionSystemTarget

  Component for celestial body object, which helps to control how PredictionSystem will display predicted motion path.
- class SceneViewDisplayManager

  Controller for HUD tools (orbits, buttons, labels display) in sceneview.
- class SimulationControl

  Main controller for gravitational motion on scene. Controls behaviour of celestial bodies, and holds global settings of gravitational simulation.
- class SphereOfInfluence

  Basic static Sphere of Influence component script, alternative to Dynamic Attractor Changing. If attached to gameobject whith no colliders, new collider will be created.
- class Vector3dDrawer

  Unity editor extension - property drawer for Vector3d type.

5.1.1 Detailed Description

Orbital motion system.

5.2 SpaceGravity2D.Inspector Namespace Reference

Classes

- class CelestialBodyEditor

  Custom editor for CelestialBody component.
- class SimulationParametersWindow

  Unity editor window for SimulationControl settings for current scene. Also manages displaying orbits path lines and other tools in sceneview window.

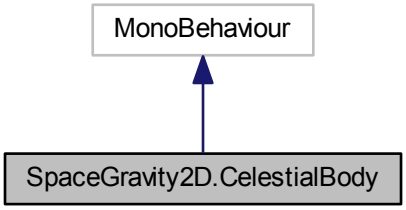
Chapter 6

Class Documentation

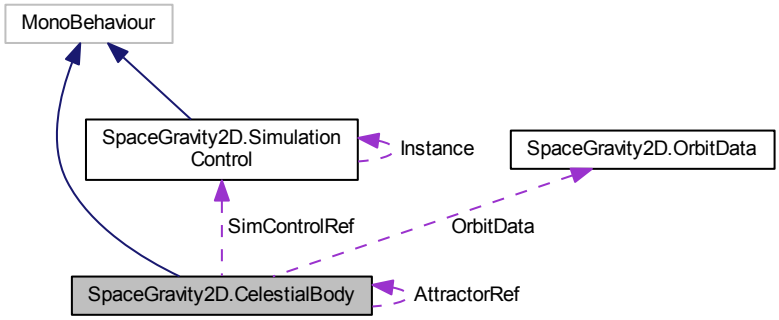
6.1 SpaceGravity2D.CelestialBody Class Reference

Component, which allows gameobject to attract other celestial bodies and be attracted by them.

Inheritance diagram for SpaceGravity2D.CelestialBody:



Collaboration diagram for SpaceGravity2D.CelestialBody:



Public Member Functions

- void FindReferences ()

  Autofind and precache required components references.
- void MakeOrbitCircle ()

  Circularize orbit. Orbit plane will not be changed.
- void MakeOrbitCircle (bool clockwise)

  Circularize orbit. Orbit plane will be unchanged.
- void SetAttractor (CelestialBody attr)

  Assign new attractor reference (or null) to this instance.
- void SetAttractor (CelestialBody attr, bool checkIsInRange, bool instant=false)

  Set new attractor at the end of frame or instantly.
- void TerminateKeplerMotion ()

  Stop Kepler motion type and return to N-body motion type at next frame.
- void FindAndSetNearestAttractor ()

  Find and assign attractor with shortest distance to this body.
- void FindAndSetMostProperAttractor ()

  Find and assign attractor with largest relative gravitational influence to this body.
- void FindAndSetBiggestAttractor ()

  Find ans assign attractor with largest mass on scene.
- void ProjectOntoEclipticPlane ()

  Make projection of position and velocity onto ecliptic plane (make 2d).
- void CalculateNewOrbitData ()

  Recalculate orbit data from current position, velocity and attractor data.
- void RefreshCurrentPositionAndVelocityFromOrbitData ()

  If internal orbit data was changed, this method will update corresponding visual parameters.
- void RotateOrbitAroundFocus (Quaternion rotation)

  Apply additional rotation to whole orbit.
- Vector3 [ ] GetOrbitPoints (int pointsCount=50, bool localSpace=false, float maxDistance=1000f)

  Get orbit points for current orbit.
- Vector3d [ ] GetOrbitPointsDouble (int pointsCount=50, bool localSpace=false, float maxDistance=1000f)

  Get orbit points for current orbit.
- void GetOrbitPointsNoAlloc (ref Vector3[ ] points, int pointsCount=50, bool localSpace=false, float max↩ Distance=1000f)

  Get orbit points for current orbit without allocation of points array.
- void GetOrbitPointsNoAlloc (ref Vector3d[ ] points, int pointsCount=50, bool localSpace=false, double max↩ Distance=1000d)

  Get orbit points for current orbit without allocation of points array.
- void AddExternalVelocity (Vector3d deltaVelocity)

  Add additional velocity to this body' velocity at the end of frame and switch to n-body motion type.
- void AddExternalVelocity (Vector3 deltaVelocity)

  Add additional velocity to this body' velocity at the end of frame and switch to n-body motion type.
- void AddExternalForce (Vector3d forceVector)

  Add force to this body's velocity. Velocity will be changed at the end of frame.
- void AddExternalForce (Vector3 forceVector)

  Add force to this body's velocity. Velocity will be changed at the end of frame.
- void SetPosition (Vector3d newPosition)

  Set new position and recalculate orbit.
- void SetPosition (Vector3 newPosition)

  Set new position and recalculate orbit.
- Vector3d GetCentralPositionAtEccentricAnomaly (double eccentricAnomaly)

Get position relative to center point of current orbit when eccentric anomaly is equal to specified value.

- Vector3d GetCentralPositionAtTrueAnomaly (double trueAnomaly)

  Get position relative to center point of current orbit when true anomaly is equal to specified value.
- Vector3d GetFocalPositionAtEccentricAnomaly (double eccentricAnomaly)

  Get position relative to focal point of current orbit when eccentric anomaly is equal to specified value.
- Vector3d GetFocalPositionAtTrueAnomaly (double trueAnomaly)

  Get position relative to focal point of current orbit when true anomaly is equal to specified value.
- Vector3d GetRelVelocityAtEccentricAnomaly (double eccentricAnomaly)

  Get velocity, relative to current attractor, when eccentric anomaly is equal to specified value.
- Vector3d GetRelVelocityAtTrueAnomaly (double trueAnomaly)

  Get velocity, relative to current attractor, when true anomaly is equal to specified value.
- void UpdateObjectOrbitDynamicParameters (double deltatime)

  Progress dynamic orbit values by specified delta time.
- bool GetAscendingNode (out Vector3 asc)

  Get local position of ascending node of current orbit.
- bool GetDescendingNode (out Vector3 desc)

  Get local position of descending node of current orbit.
- Vector3 GetVelocityPlaneNormal ()

  Gets normal vector to plane, which is defined by body position and velocity vector. If orbit is valid, velocity normal is equal to orbit normal, otherwise velocity normal will be calculated as perpendicular to velocity and ecliptic up vector.

Public Attributes

- SimulationControl SimControlRef

  Reference to main controller. Should never be Null.
- double Mass = 1f

  Body mass value. Should not be less than 1.
- double MaxAttractionRange = double.PositiveInfinity

  Maximum range of attraction force in world units.
- CelestialBody AttractorRef

  Attractor body reference.
- Vector3d Velocity

  World space velocity vector of the body.
- bool IsFixedPosition

  Is currently position fixed in place (relative to current attractor).
- bool IsKeplerMotion = true

  Is rail motion type active at this frame. If false, then N-body motion type will be active. Don't change this manually. modify UseKeplerMotion instead.
- bool UseKeplerMotion

  Motion type switch. Switch kepler and N-body motion type.
- float SearchAttractorInterval = 1.0f

  Interval for continious attractor search process in seconds.
- Vector3d AdditionalVelocity

  The additional velocity, which was added in current frame. Value of velocity will be added to main velocity once per frame.
- OrbitData OrbitData = new OrbitData()

  Current internal orbit state data.

Properties

- SimulationControl simControlRef `[get, set]`

  Reference to main controller. Should never be Null.
- Vector3d position `[get, set]`

  World position vector with double precision.
- Vector3d Position `[get, set]`

  World position vector with double precision.
- Vector3d focalPosition `[get, set]`

  Position relative to attractor.
- Vector3d FocalPosition `[get, set]`

  Position relative to attractor.
- Vector3d centralPosition `[get]`

  Position relative to orbit center.
- Vector3d CentralPosition `[get]`

  Position relative to orbit center.
- double mass `[get, set]`

  Body mass value. Should not be less than 1.
- double MG `[get]`

  Gravitational parameter of body [Mass ∗ GravConst].
- double maxAttractorRange `[get, set]`

  Maximum range of attraction force in world units.
- CelestialBody attractor `[get, set]`

  Attractor body reference.
- Vector3d velocity `[get, set]`

  World space velocity vector of the body.
- Vector3d relativeVelocity `[get, set]`

  World space velocity, relative to current attractor.
- Vector3d RelativeVelocity `[get, set]`

  World space velocity, relative to current attractor.
- Vector3d relativePosition `[get, set]`

  Position, relative to current attractor, with double precision.
- Vector3d RelativePosition `[get, set]`

  Position, relative to current attractor, with double precision.
- bool isFixedPosition `[get, set]`

  Is currently position fixed in place (relative to current attractor).
- bool isKeplerMotion `[get, set]`

  Is rail motion type active at this frame. If false, then N-body motion type will be active.
- bool useKeplerMotion `[get, set]`

  Motion type switch. Switch kepler and N-body motion type.
- bool isAttractorSearchActive `[get, set]`

  Dynamic search of most proper attractor toggle.
- bool IsAttractorSearchActive `[get, set]`

  Dynamic search of most proper attractor toggle. Gets the current state of attractor search process. Sets attractor search state (enabled or disabled).
- float searchAttractorInterval `[get, set]`

  Interval for continious attractor search process in seconds.
- Vector3d additionalVelocity `[get, set]`

  The additional velocity, which was added in current frame. Value of velocity will be added to main velocity once per frame.
- OrbitData orbitData `[get, set]`

Current internal orbit state data.

- Vector3d orbitFocusPoint `[get]`

  Current world position of orbit focus (attractor position).
- Vector3d OrbitFocusPoint `[get]`

  Current world position of orbit focus (attractor position).
- Vector3d orbitCenterPoint `[get]`

  World position of orbit center.
- Vector3d OrbitCenterPoint `[get]`

  World position of orbit center.
- Vector3d orbitPeriapsisPoint `[get]`

  World position of periapsis orbit point.
- Vector3d OrbitPeriapsisPoint `[get]`

  World position of lowest orbit point.
- Vector3d orbitApoapsisPoint `[get]`

  World position of highest orbit point.
- Vector3d OrbitApoapsisPoint `[get]`

  World position of highest orbit point.
- bool isValidOrbit `[get]`

  Is current state of orbit errorless.
- bool IsValidOrbit `[get]`

  Is current state of orbit errorless.
- Vector3d centerOfMass `[get]`

  World position of center of mass of body and current attractor.
- Vector3d CenterOfMass `[get]`

  World position of center of mass of body and current attractor.
- double eccentricity `[get, set]`

  Eccentricity of current orbit.
- double Eccentricity `[get, set]`

  Eccentricity of current orbit.
- double trueAnomaly `[get, set]`

  True anomaly of current orbit.
- double TrueAnomaly `[get, set]`

  True anomaly of current orbit.
- double eccentricAnomaly `[get, set]`

  Eccentric anomaly of current orbit in radians.
- double EccentricAnomaly `[get, set]`

  Eccentric anomaly of current orbit in radians.
- double meanAnomaly `[get, set]`

  Mean anomaly of current orbit in radians.
- double MeanAnomaly `[get, set]`

  Mean anomaly of current orbit in radians.

Events

- static Action< CelestialBody > OnBodyCreatedEvent

  Static event, which was used to register creation of celestial body in SimulationControl.
- static Action< CelestialBody > OnBodyDestroyedEvent

  Static event, which was used to register creation of celestial body in SimulationControl.
- Action OnDestroyedEvent

  Occuring when body was destroyed.
- Action OnEnabledEvent

  Occuring when body was created or enabled.
- Action OnDisabledEvent

  Occuring when body was destroyed or disabled.

6.1.1 Detailed Description

Component, which allows gameobject to attract other celestial bodies and be attracted by them.

6.1.2 Member Function Documentation

6.1.2.1 AddExternalForce() [1/2]

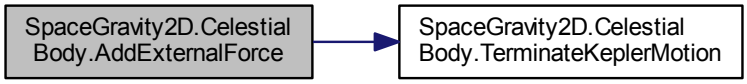
```
void SpaceGravity2D.CelestialBody.AddExternalForce (
            Vector3d forceVector )
```

Add force to this body's velocity. Velocity will be changed at the end of frame.

Parameters

forceVector	Force direction and magnitude vector.

Here is the call graph for this function:



Here is the caller graph for this function:

6.1.2.2 AddExternalForce() [2/2]

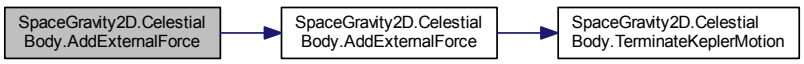
```
void SpaceGravity2D.CelestialBody.AddExternalForce (
            Vector3 forceVector )
```

Add force to this body's velocity. Velocity will be changed at the end of frame.

Parameters

forceVector	Force direction and magnitude vector.

Here is the call graph for this function:



6.1.2.3 AddExternalVelocity() [1/2]

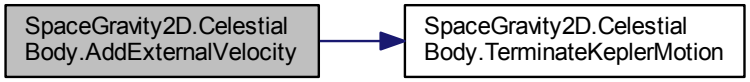
```
void SpaceGravity2D.CelestialBody.AddExternalVelocity (
            Vector3d deltaVelocity )
```

Add additional velocity to this body' velocity at the end of frame and switch to n-body motion type.

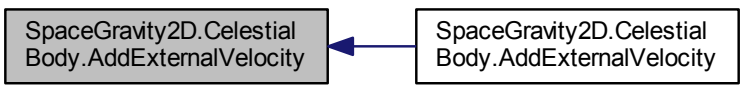
Parameters

deltaVelocity	Additional velocity.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.4 AddExternalVelocity() [2/2]

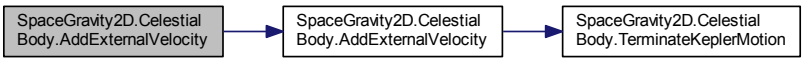
```
void SpaceGravity2D.CelestialBody.AddExternalVelocity (
            Vector3 deltaVelocity )
```

Add additional velocity to this body' velocity at the end of frame and switch to n-body motion type.

Parameters

deltaVelocity	Additional velocity.

Here is the call graph for this function:

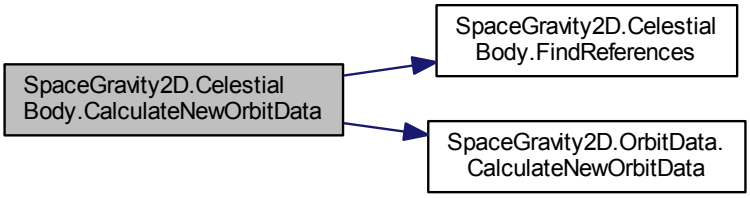


6.1.2.5 CalculateNewOrbitData()

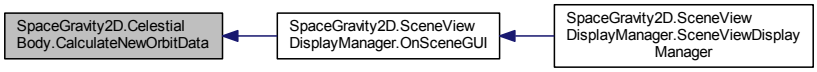
```
void SpaceGravity2D.CelestialBody.CalculateNewOrbitData ( )
```

Recalculate orbit data from current position, velocity and attractor data.

Should be called after manual change of transform of body or attractor, or other visual parameters to update internal orbit state. Here is the call graph for this function:



Here is the caller graph for this function:

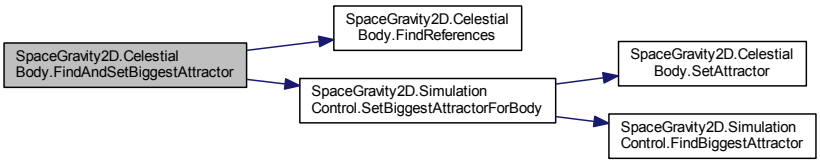


6.1.2.6 FindAndSetBiggestAttractor()

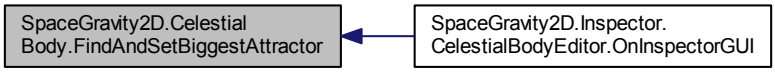
```
void SpaceGravity2D.CelestialBody.FindAndSetBiggestAttractor ( )
```

Find ans assign attractor with largest mass on scene.

Here is the call graph for this function:



Here is the caller graph for this function:

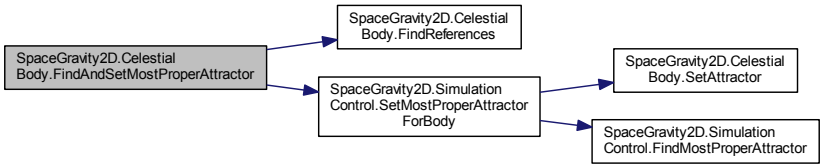


6.1.2.7 FindAndSetMostProperAttractor()

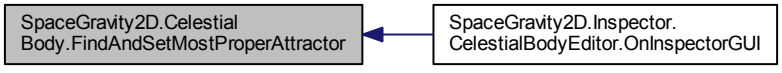
```
void SpaceGravity2D.CelestialBody.FindAndSetMostProperAttractor ( )
```

Find and assign attractor with largest relative gravitational influence to this body.

Here is the call graph for this function:



Here is the caller graph for this function:

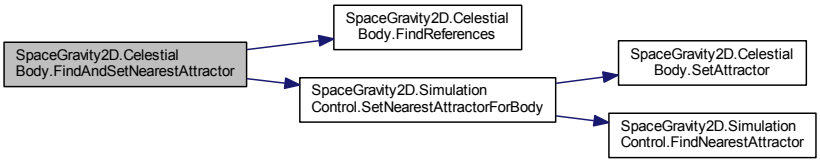


6.1.2.8 FindAndSetNearestAttractor()

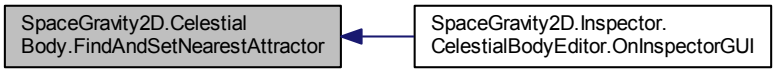
```
void SpaceGravity2D.CelestialBody.FindAndSetNearestAttractor ( )
```

Find and assign attractor with shortest distance to this body.

Here is the call graph for this function:



Here is the caller graph for this function:

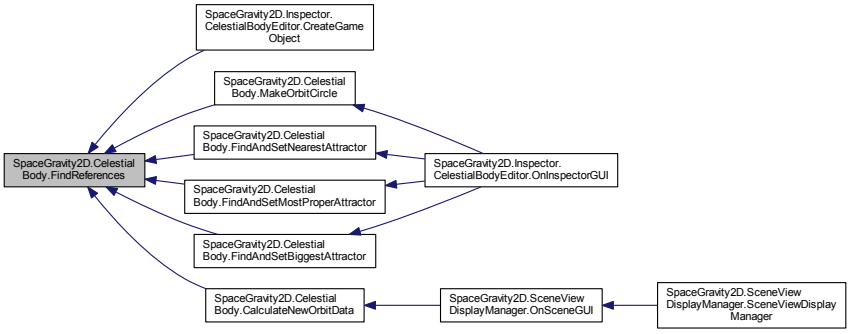


6.1.2.9 FindReferences()

```
void SpaceGravity2D.CelestialBody.FindReferences ( )
```

Autofind and precache required components references.

Here is the caller graph for this function:



6.1.2.10 GetAscendingNode()

```
bool SpaceGravity2D.CelestialBody.GetAscendingNode (
            out Vector3 asc )
```

Get local position of ascending node of current orbit.

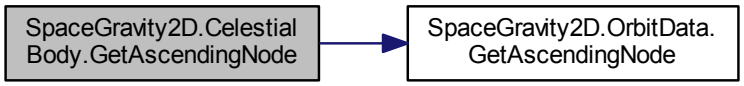
Parameters

asc	Resulting point.

Returns

Operation success status.

Here is the call graph for this function:



6.1.2.11 GetCentralPositionAtEccentricAnomaly()

```
Vector3d SpaceGravity2D.CelestialBody.GetCentralPositionAtEccentricAnomaly (
            double eccentricAnomaly )
```

Get position relative to center point of current orbit when eccentric anomaly is equal to specified value.

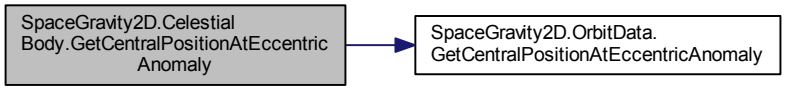
Parameters

eccentricAnomaly	Position on current orbit, determined by eccentric anomaly orbital parameter.

Returns

Position, relative to orbit center.

Here is the call graph for this function:



6.1.2.12 GetCentralPositionAtTrueAnomaly()

```
Vector3d SpaceGravity2D.CelestialBody.GetCentralPositionAtTrueAnomaly (
            double trueAnomaly )
```

Get position relative to center point of current orbit when true anomaly is equal to specified value.

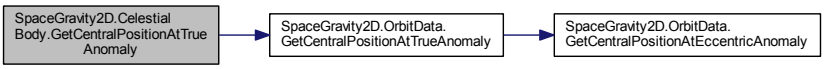
Parameters

trueAnomaly	Position on current orbit, determined by true anomaly orbital parameter.

Returns

Position, relative to orbit center.

Here is the call graph for this function:



6.1.2.13 GetDescendingNode()

```
bool SpaceGravity2D.CelestialBody.GetDescendingNode (
            out Vector3 desc )
```

Get local position of descending node of current orbit.

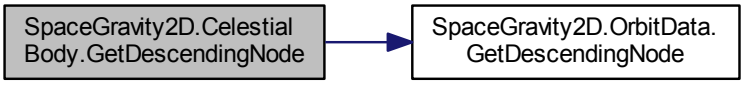
Parameters

desc	Resulting point.

Returns

Operation success status.

Here is the call graph for this function:



6.1.2.14 GetFocalPositionAtEccentricAnomaly()

```
Vector3d SpaceGravity2D.CelestialBody.GetFocalPositionAtEccentricAnomaly (
            double eccentricAnomaly )
```

Get position relative to focal point of current orbit when eccentric anomaly is equal to specified value.

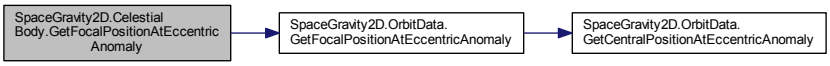
Parameters

eccentricAnomaly	Position on current orbit, determined by eccentric anomaly orbital parameter.

Returns

Position, relative to orbit focus.

Here is the call graph for this function:



6.1.2.15 GetFocalPositionAtTrueAnomaly()

```
Vector3d SpaceGravity2D.CelestialBody.GetFocalPositionAtTrueAnomaly (
            double trueAnomaly )
```

Get position relative to focal point of current orbit when true anomaly is equal to specified value.

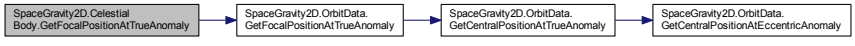
Parameters

trueAnomaly	Position on current orbit, determined by true anomaly orbital parameter.

Returns

Position, relative to orbit focus.

Here is the call graph for this function:



6.1.2.16 GetOrbitPoints()

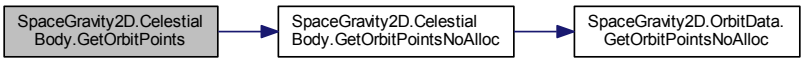
```
Vector3 [] SpaceGravity2D.CelestialBody.GetOrbitPoints (
            int pointsCount = 50,
            bool localSpace = false,
            float maxDistance = 1000f )
```

Get orbit points for current orbit.

Returns

Orbit curve points array.

Here is the call graph for this function:



6.1.2.17 GetOrbitPointsDouble()

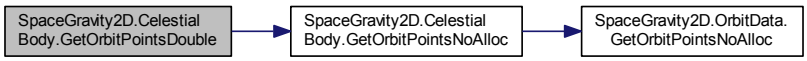
```
Vector3d [] SpaceGravity2D.CelestialBody.GetOrbitPointsDouble (
            int pointsCount = 50,
            bool localSpace = false,
            float maxDistance = 1000f )
```

Get orbit points for current orbit.

Returns

Orbit curve points array.

Here is the call graph for this function:



6.1.2.18 GetOrbitPointsNoAlloc() [1/2]

```
void SpaceGravity2D.CelestialBody.GetOrbitPointsNoAlloc (
            ref Vector3 [] points,
            int pointsCount = 50,
            bool localSpace = false,
            float maxDistance = 1000f )
```

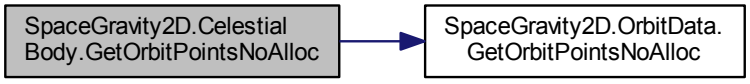
Get orbit points for current orbit without allocation of points array.

Note: array allocation may sometimes occur, if specified array is null or lenght is not equal to target points count. And target points count not always equal to pointsCount parameter, because sometimes orbit curve doesn't require maximym points count.

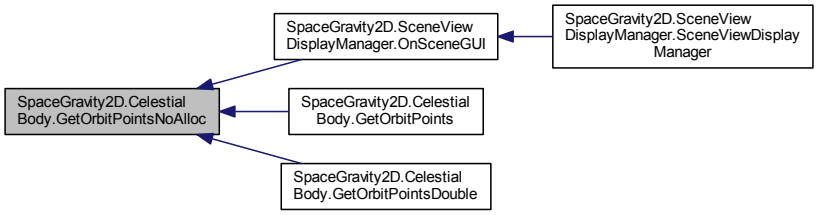
Parameters

points	Array of orbut curve points.
pointsCount	Maximum curve points count.
localSpace	Is curve centered to this body, or to world space.
maxDistance	Max distance for curve points.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.19 GetOrbitPointsNoAlloc() [2/2]

```
void SpaceGravity2D.CelestialBody.GetOrbitPointsNoAlloc (
            ref Vector3d [] points,
            int pointsCount = 50,
            bool localSpace = false,
            double maxDistance = 1000d )
```

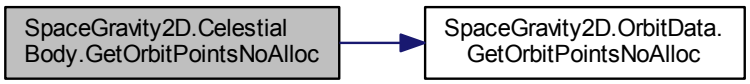
Get orbit points for current orbit without allocation of points array.

Note: array allocation may sometimes occur, if specified array is null or lenght is not equal to target points count. And target points count not always equal to pointsCount parameter, because sometimes orbit curve doesn't require maximym points count.

Parameters

points	Array of orbut curve points.
pointsCount	Maximum curve points count.
localSpace	Is curve centered to this body, or to world space.
maxDistance	Max distance for curve points.

Here is the call graph for this function:



6.1.2.20 GetRelVelocityAtEccentricAnomaly()

```
Vector3d SpaceGravity2D.CelestialBody.GetRelVelocityAtEccentricAnomaly (
            double eccentricAnomaly )
```

Get velocity, relative to current attractor, when eccentric anomaly is equal to specified value.

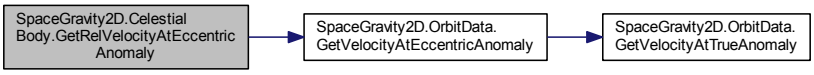
Parameters

eccentricAnomaly	Velocity, relative to attractor at orbit point, determined by eccentric anomaly.

Returns

Relative velocity.

Here is the call graph for this function:



6.1.2.21 GetRelVelocityAtTrueAnomaly()

```
Vector3d SpaceGravity2D.CelestialBody.GetRelVelocityAtTrueAnomaly (
            double trueAnomaly )
```

Get velocity, relative to current attractor, when true anomaly is equal to specified value.

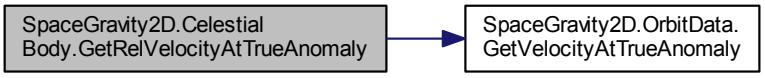
Parameters

trueAnomaly	Velocity, relative to attractor at orbit point, determined by true anomaly.

Returns

Relative velocity.

Here is the call graph for this function:



6.1.2.22 GetVelocityPlaneNormal()

```
Vector3 SpaceGravity2D.CelestialBody.GetVelocityPlaneNormal ( )
```

Gets normal vector to plane, which is defined by body position and velocity vector. If orbit is valid, velocity normal is equal to orbit normal, otherwise velocity normal will be calculated as perpendicular to velocity and ecliptic up vector.

Returns

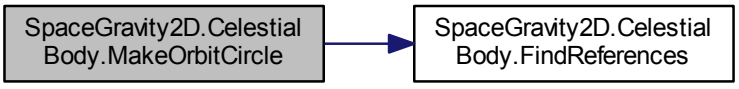
Unit vector, perpendicular to velocity.

6.1.2.23 MakeOrbitCircle() [1/2]

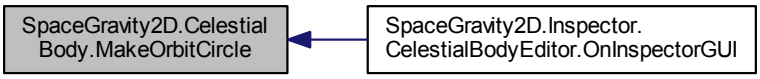
```
void SpaceGravity2D.CelestialBody.MakeOrbitCircle ( )
```

Circularize orbit. Orbit plane will not be changed.

Behaviour update: resulting orbit will be always same orientation, as before calling this method (not reversing velocity). Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2.24 MakeOrbitCircle() [2/2]

```
void SpaceGravity2D.CelestialBody.MakeOrbitCircle (
            bool clockwise )
```

Circularize orbit. Orbit plane will be unchanged.

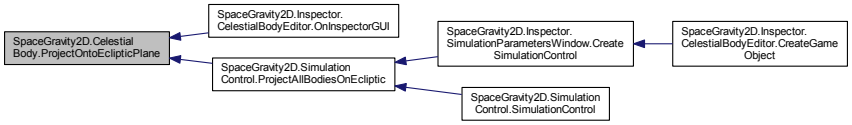
Here is the call graph for this function:

6.1.2.25 ProjectOntoEclipticPlane()

```
void SpaceGravity2D.CelestialBody.ProjectOntoEclipticPlane ( )
```

Make projection of position and velocity onto ecliptic plane (make 2d).

If call this method every frame, simulation will be restricted in 2d plane, which may be usefull. There is such setting in SpaceGravity2DWindow. Here is the caller graph for this function:



6.1.2.26 RefreshCurrentPositionAndVelocityFromOrbitData()

```
void SpaceGravity2D.CelestialBody.RefreshCurrentPositionAndVelocityFromOrbitData ( )
```

If internal orbit data was changed, this method will update corresponding visual parameters.

Call this method if manually changing orbit data, while motion type is KeplerMotion. (N-body doesn't require refreshing, because orbit data is recalulating every frame anyway).

6.1.2.27 RotateOrbitAroundFocus()

```
void SpaceGravity2D.CelestialBody.RotateOrbitAroundFocus (
            Quaternion rotation )
```

Apply additional rotation to whole orbit.

Parameters

rotation	Rotation to add.

For example, quaternion.euler(15,0,0) will rotate orbit for 15 degrees around x axis. Here is the call graph for this function:

6.1.2.28 SetAttractor() [1/2]

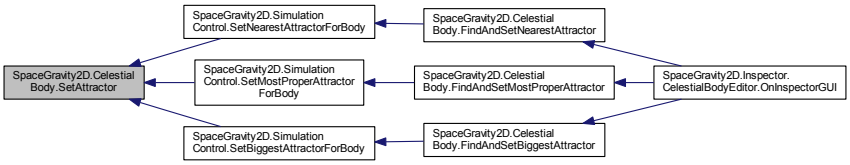
```
void SpaceGravity2D.CelestialBody.SetAttractor (
            CelestialBody attr )
```

Assign new attractor reference (or null) to this instance.

Parameters

attr	Attractor instance or null.

Here is the caller graph for this function:



6.1.2.29 SetAttractor() [2/2]

```
void SpaceGravity2D.CelestialBody.SetAttractor (
            CelestialBody attr,
            bool checkIsInRange,
            bool instant = false )
```

Set new attractor at the end of frame or instantly.

6.1.2.30 SetPosition() [1/2]

```
void SpaceGravity2D.CelestialBody.SetPosition (
            Vector3d newPosition )
```

Set new position and recalculate orbit.

Parameters

newPosition	New world position.

Here is the caller graph for this function:

6.1.2.31 SetPosition() [2/2]

```
void SpaceGravity2D.CelestialBody.SetPosition (
            Vector3 newPosition )
```

Set new position and recalculate orbit.

Parameters

newPosition	New world position.

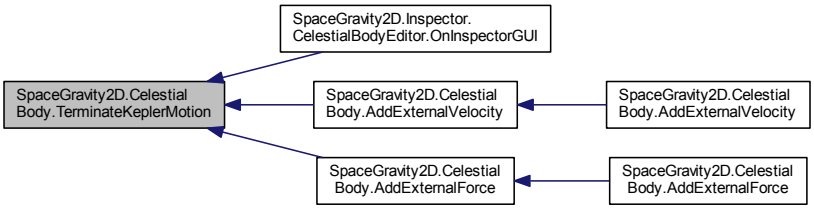
Here is the call graph for this function:

6.1.2.32 TerminateKeplerMotion()

```
void SpaceGravity2D.CelestialBody.TerminateKeplerMotion ( )
```

Stop Kepler motion type and return to N-body motion type at next frame.

Here is the caller graph for this function:



6.1.2.33 UpdateObjectOrbitDynamicParameters()

```
void SpaceGravity2D.CelestialBody.UpdateObjectOrbitDynamicParameters (
            double deltatime )
```

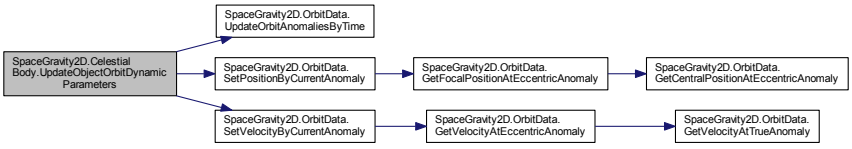
Progress dynamic orbit values by specified delta time.

If current motion type is Kepler Motion, then this method is called every frame from global scene controller and such orbit parameters, as mean anomaly, true anomaly and eccentric anomaly will be changed according to specified delta time. And then new position and velocity values will be calculated from anomalies.

Parameters

deltatime	Progress time.

Here is the call graph for this function:



6.1.3 Member Data Documentation

6.1.3.1 AdditionalVelocity

```
Vector3d SpaceGravity2D.CelestialBody.AdditionalVelocity
```

The additional velocity, which was added in current frame. Value of velocity will be added to main velocity once per frame.

Used to bufferize external velocity change, so orbit can be recalculated only once, if multiple external changes occured during single frame.

6.1.3.2 AttractorRef

```
CelestialBody SpaceGravity2D.CelestialBody.AttractorRef
```

Attractor body reference.

Not required only if motion type is N-body. Used for calculating orbit state in OrbitData. If OrbitData can't be calculated, Kepler motion type and orbit display is not possible.

6.1.3.3 IsFixedPosition

```
bool SpaceGravity2D.CelestialBody.IsFixedPosition
```

Is currently position fixed in place (relative to current attractor).

6.1.3.4 IsKeplerMotion

```
bool SpaceGravity2D.CelestialBody.IsKeplerMotion = true
```

Is rail motion type active at this frame. If false, then N-body motion type will be active. Don't change this manually. modify UseKeplerMotion instead.

6.1.3.5 Mass

```
double SpaceGravity2D.CelestialBody.Mass = 1f
```

Body mass value. Should not be less than 1.

If mass is bigger than attractor mass treshold, this body will become attractor. Mass value should always be larger than 1, because of division by zero (floating point values between 0 and 1 are ok, but it is easier to set and maintain min value 1).

6.1.3.6 MaxAttractionRange

```
double SpaceGravity2D.CelestialBody.MaxAttractionRange = double.PositiveInfinity
```

Maximum range of attraction force in world units.

6.1.3.7 OrbitData

```
OrbitData SpaceGravity2D.CelestialBody.OrbitData = new OrbitData()
```

Current internal orbit state data.

OrbitData contains all orbit parameters for Kepler motion type and also is used to display current orbit. N-body motion type doesn't require OrbitData. OrbitData will be updated every frame only if needed. So, if current motion type is N-body, and orbit path display is turned on, then OrbitData still will be updated every frame.

6.1.3.8 SearchAttractorInterval

```
float SpaceGravity2D.CelestialBody.SearchAttractorInterval = 1.0f
```

Interval for continious attractor search process in seconds.

Most proper attractor search process is quite expensive for performance, so it should not be executed every frame. This interval determines how often it should be performed.

6.1.3.9 SimControlRef

```
SimulationControl SpaceGravity2D.CelestialBody.SimControlRef
```

Reference to main controller. Should never be Null.

If celestial body is created on scene, where Simulation control is not exist, then new default simulation control will be created. Otherwise existing simulation control will be found and placed here as reference automatically.

6.1.3.10 UseKeplerMotion

```
bool SpaceGravity2D.CelestialBody.UseKeplerMotion
```

Motion type switch. Switch kepler and N-body motion type.

6.1.3.11 Velocity

```
Vector3d SpaceGravity2D.CelestialBody.Velocity
```

World space velocity vector of the body.

6.1.4 Property Documentation

6.1.4.1 additionalVelocity

```
Vector3d SpaceGravity2D.CelestialBody.additionalVelocity  [get], [set]
```

The additional velocity, which was added in current frame. Value of velocity will be added to main velocity once per frame.

6.1.4.2 attractor

```
CelestialBody SpaceGravity2D.CelestialBody.attractor  [get], [set]
```

Attractor body reference.

6.1.4.3 centerOfMass

```
Vector3d SpaceGravity2D.CelestialBody.centerOfMass [get]
```

World position of center of mass of body and current attractor.

6.1.4.4 CenterOfMass

```
Vector3d SpaceGravity2D.CelestialBody.CenterOfMass [get]
```

World position of center of mass of body and current attractor.

6.1.4.5 centralPosition

```
Vector3d SpaceGravity2D.CelestialBody.centralPosition [get]
```

Position relative to orbit center.

6.1.4.6 CentralPosition

```
Vector3d SpaceGravity2D.CelestialBody.CentralPosition [get]
```

Position relative to orbit center.

Note: orbit center is not equal to attractor position (if Eccentricity > 0)

6.1.4.7 eccentricAnomaly

```
double SpaceGravity2D.CelestialBody.eccentricAnomaly [get], [set]
```

Eccentric anomaly of current orbit in radians.

6.1.4.8 EccentricAnomaly

```
double SpaceGravity2D.CelestialBody.EccentricAnomaly [get], [set]
```

Eccentric anomaly of current orbit in radians.

6.1.4.9 eccentricity

```
double SpaceGravity2D.CelestialBody.eccentricity [get], [set]
```

Eccentricity of current orbit.

6.1.4.10 Eccentricity

```
double SpaceGravity2D.CelestialBody.Eccentricity  [get], [set]
```

Eccentricity of current orbit.

6.1.4.11 focalPosition

```
Vector3d SpaceGravity2D.CelestialBody.focalPosition  [get], [set]
```

Position relative to attractor.

6.1.4.12 FocalPosition

```
Vector3d SpaceGravity2D.CelestialBody.FocalPosition  [get], [set]
```

Position relative to attractor.

6.1.4.13 isAttractorSearchActive

```
bool SpaceGravity2D.CelestialBody.isAttractorSearchActive  [get], [set]
```

Dynamic search of most proper attractor toggle.

6.1.4.14 IsAttractorSearchActive

```
bool SpaceGravity2D.CelestialBody.IsAttractorSearchActive  [get], [set]
```

Dynamic search of most proper attractor toggle. Gets the current state of attractor search process. Sets attractor search state (enabled or disabled).

6.1.4.15 isFixedPosition

```
bool SpaceGravity2D.CelestialBody.isFixedPosition  [get], [set]
```

Is currently position fixed in place (relative to current attractor).

6.1.4.16 isKeplerMotion

```
bool SpaceGravity2D.CelestialBody.isKeplerMotion  [get], [set]
```

Is rail motion type active at this frame. If false, then N-body motion type will be active.

6.1.4.17 isValidOrbit

```
bool SpaceGravity2D.CelestialBody.isValidOrbit  [get]
```

Is current state of orbit errorless.

6.1.4.18 IsValidOrbit

```
bool SpaceGravity2D.CelestialBody.IsValidOrbit  [get]
```

Is current state of orbit errorless.

6.1.4.19 mass

```
double SpaceGravity2D.CelestialBody.mass  [get], [set]
```

Body mass value. Should not be less than 1.

6.1.4.20 maxAttractorRange

```
double SpaceGravity2D.CelestialBody.maxAttractorRange  [get], [set]
```

Maximum range of attraction force in world units.

6.1.4.21 meanAnomaly

```
double SpaceGravity2D.CelestialBody.meanAnomaly  [get], [set]
```

Mean anomaly of current orbit in radians.

6.1.4.22 MeanAnomaly

```
double SpaceGravity2D.CelestialBody.MeanAnomaly  [get], [set]
```

Mean anomaly of current orbit in radians.

6.1.4.23 MG

```
double SpaceGravity2D.CelestialBody.MG  [get]
```

Gravitational parameter of body [Mass $*$ GravConst].

6.1.4.24 orbitApoapsisPoint

```
Vector3d SpaceGravity2D.CelestialBody.orbitApoapsisPoint  [get]
```

World position of highest orbit point.

6.1.4.25 OrbitApoapsisPoint

```
Vector3d SpaceGravity2D.CelestialBody.OrbitApoapsisPoint  [get]
```

World position of highest orbit point.

6.1.4.26 orbitCenterPoint

```
Vector3d SpaceGravity2D.CelestialBody.orbitCenterPoint  [get]
```

World position of orbit center.

6.1.4.27 OrbitCenterPoint

```
Vector3d SpaceGravity2D.CelestialBody.OrbitCenterPoint  [get]
```

World position of orbit center.

6.1.4.28 orbitData

```
OrbitData SpaceGravity2D.CelestialBody.orbitData  [get], [set]
```

Current internal orbit state data.

6.1.4.29 orbitFocusPoint

```
Vector3d SpaceGravity2D.CelestialBody.orbitFocusPoint  [get]
```

Current world position of orbit focus (attractor position).

6.1.4.30 OrbitFocusPoint

```
Vector3d SpaceGravity2D.CelestialBody.OrbitFocusPoint  [get]
```

Current world position of orbit focus (attractor position).

6.1.4.31 orbitPeriapsisPoint

```
Vector3d SpaceGravity2D.CelestialBody.orbitPeriapsisPoint  [get]
```

World position of periapsis orbit point.

6.1.4.32 OrbitPeriapsisPoint

```
Vector3d SpaceGravity2D.CelestialBody.OrbitPeriapsisPoint  [get]
```

World position of lowest orbit point.

6.1.4.33 position

```
Vector3d SpaceGravity2D.CelestialBody.position  [get], [set]
```

World position vector with double precision.

6.1.4.34 Position

```
Vector3d SpaceGravity2D.CelestialBody.Position  [get], [set]
```

World position vector with double precision.

6.1.4.35 relativePosition

```
Vector3d SpaceGravity2D.CelestialBody.relativePosition  [get], [set]
```

Position, relative to current attractor, with double precision.

6.1.4.36 RelativePosition

```
Vector3d SpaceGravity2D.CelestialBody.RelativePosition  [get], [set]
```

Position, relative to current attractor, with double precision.

6.1.4.37 relativeVelocity

```
Vector3d SpaceGravity2D.CelestialBody.relativeVelocity  [get], [set]
```

World space velocity, relative to current attractor.

6.1.4.38 RelativeVelocity

```
Vector3d SpaceGravity2D.CelestialBody.RelativeVelocity  [get], [set]
```

World space velocity, relative to current attractor.

6.1.4.39 searchAttractorInterval

```
float SpaceGravity2D.CelestialBody.searchAttractorInterval  [get], [set]
```

Interval for continious attractor search process in seconds.

6.1.4.40 simControlRef

```
SimulationControl SpaceGravity2D.CelestialBody.simControlRef  [get], [set]
```

Reference to main controller. Should never be Null.

6.1.4.41 trueAnomaly

```
double SpaceGravity2D.CelestialBody.trueAnomaly  [get], [set]
```

True anomaly of current orbit.

6.1.4.42 TrueAnomaly

```
double SpaceGravity2D.CelestialBody.TrueAnomaly  [get], [set]
```

True anomaly of current orbit.

6.1.4.43 useKeplerMotion

```
bool SpaceGravity2D.CelestialBody.useKeplerMotion  [get], [set]
```

Motion type switch. Switch kepler and N-body motion type.

6.1.4.44 velocity

```
Vector3d SpaceGravity2D.CelestialBody.velocity  [get], [set]
```

World space velocity vector of the body.

6.1.5 Event Documentation

6.1.5.1 OnBodyCreatedEvent

```
Action<CelestialBody> SpaceGravity2D.CelestialBody.OnBodyCreatedEvent  [static]
```

Static event, which was used to register creation of celestial body in SimulationControl.

6.1.5.2 OnBodyDestroyedEvent

```
Action<CelestialBody> SpaceGravity2D.CelestialBody.OnBodyDestroyedEvent  [static]
```

Static event, which was used to register creation of celestial body in SimulationControl.

6.1.5.3 OnDestroyedEvent

```
Action SpaceGravity2D.CelestialBody.OnDestroyedEvent
```

Occuring when body was destroyed.

6.1.5.4 OnDisabledEvent

```
Action SpaceGravity2D.CelestialBody.OnDisabledEvent
```

Occuring when body was destroyed or disabled.

6.1.5.5 OnEnabledEvent

```
Action SpaceGravity2D.CelestialBody.OnEnabledEvent
```

Occuring when body was created or enabled.

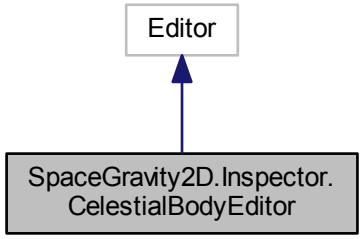
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/CelestialBody.cs

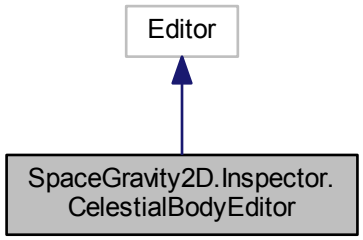
6.2 SpaceGravity2D.Inspector.CelestialBodyEditor Class Reference

Custom editor for CelestialBody component.

Inheritance diagram for SpaceGravity2D.Inspector.CelestialBodyEditor:



Collaboration diagram for SpaceGravity2D.Inspector.CelestialBodyEditor:



Public Member Functions

- override void OnInspectorGUI ()

Static Public Member Functions

- static void CreateGameObject ()
  Create new CelestialObject on scene.

6.2.1 Detailed Description

Custom editor for CelestialBody component.

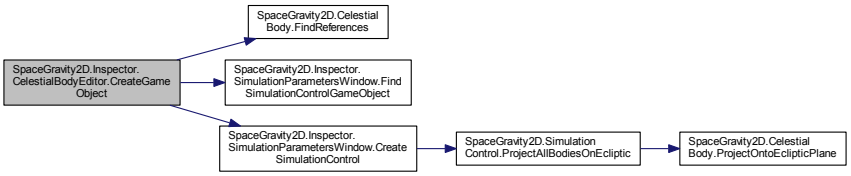
6.2.2 Member Function Documentation

6.2.2.1 CreateGameObject()

```
static void SpaceGravity2D.Inspector.CelestialBodyEditor.CreateGameObject ( )  [static]
```

Create new CelestialObject on scene.

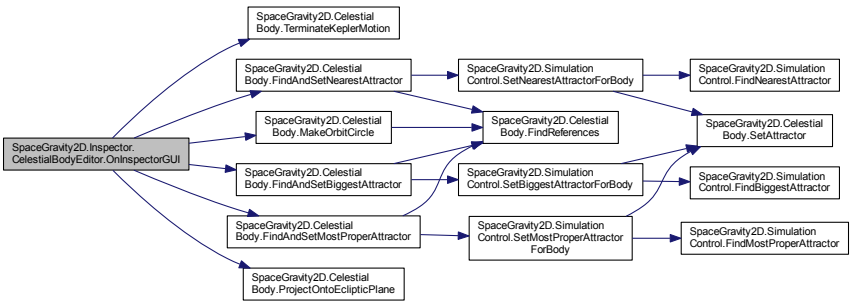
Here is the call graph for this function:



6.2.2.2 OnInspectorGUI()

```
override void SpaceGravity2D.Inspector.CelestialBodyEditor.OnInspectorGUI ( )
```

Here is the call graph for this function:



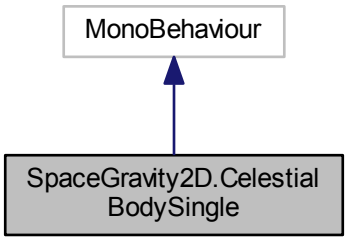
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Editor/CelestialBodyEditor.cs

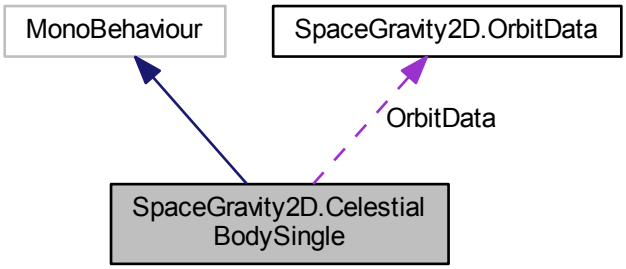
6.3 SpaceGravity2D.CelestialBodySingle Class Reference

Component for standalone static orbiting body, which is independent from SimulationControl.

Inheritance diagram for SpaceGravity2D.CelestialBodySingle:



Collaboration diagram for SpaceGravity2D.CelestialBodySingle:



Public Member Functions

- void CreateNewOrbitFromPositionAndVelocity (Vector3 relativePosition, Vector3 velocity)

  Updates OrbitData from new body position and velocity vectors.
- void ForceUpdateViewFromInternalState ()

  Forces the update of body position, and velocity handler from OrbitData. Call this method after any direct changing of OrbitData.
- void ForceUpdateOrbitData ()

  Forces the update of internal orbit data from current world positions of body, attractor settings and velocityHandle.
- void SetAutoCircleOrbit ()

  Change orbit velocity vector to match circular orbit.

Public Attributes

- Transform AttractorObjectRef

  Reference to attractor transform.
- float AttractorMass = 1000

  Attractor's mass. Should be bigger than 1.
- float MaxOrbitWorldUnitsDistance = 100f

  Max distance for display orbit in world units.
- float GravitationalConstant = 0.001f

  Gravitational constant.
- Transform VelocityHandleRef

  Reference to velocity handle object. Assign object and use it as velocity control handle in scene view.
- float VelocityMlt = 1f

  Multiplier for velocity;
- float TimeScale = 1f

  Motion total speed setting.
- OrbitData OrbitData = new OrbitData()

  The orbit data. Internal state of orbit.
- int OrbitPointsCount = 50

  Max display orbit points count. More points - better precision.
- LineRenderer LineRendererRef

  Reference to orbit linerenderer. Required only if orbit display is used.
- bool LockOrbitEditing = false

  Disable continious editing orbit in update loop, if you don't need it.

Properties

- Transform attractorObject `[get]`

  Reference to attractor transform.
- float attractorMass `[get]`

  Attractor's mass. Should be bigger than 1.
- float maxDistForHyperbolicCase `[get]`

  Max distance for display orbit.
- float G `[get]`

  Gravitational constant.
- Transform velocityHandle `[get]`

  Reference to velocity handle object. Assign object and use it as velocity control handle in scene view.
- float velocityMlt `[get]`

  Multiplier for velocity;
- OrbitData orbitData `[get]`

  The orbit data. Internal state of orbit.
- int orbitPointsCount `[get]`
- LineRenderer linerend `[get]`

  Reference to orbit linerenderer.

6.3.1 Detailed Description

Component for standalone static orbiting body, which is independent from SimulationControl.

This component is designed for situations, when only static orbit motion is required, and no any interactions with other bodies. Attractor parameters placed inside this component, so any gameobject can play role of attractor.

6.3.2 Member Function Documentation

6.3.2.1 CreateNewOrbitFromPositionAndVelocity()

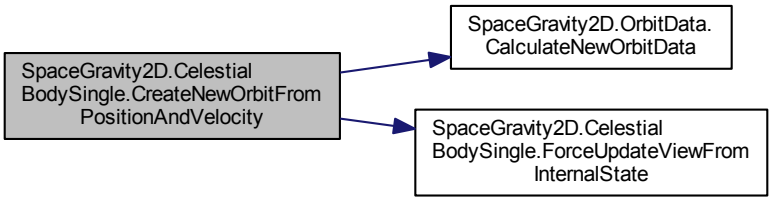
```
void SpaceGravity2D.CelestialBodySingle.CreateNewOrbitFromPositionAndVelocity (
            Vector3 relativePosition,
            Vector3 velocity )
```

Updates OrbitData from new body position and velocity vectors.

Parameters

relativePosition	The relative position.
velocity	The relative velocity.

This method can be useful to assign new position of body by script. Or you can directly change OrbitData state and then manually update view. Here is the call graph for this function:

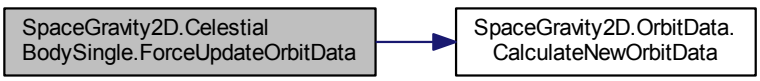


6.3.2.2 ForceUpdateOrbitData()

```
void SpaceGravity2D.CelestialBodySingle.ForceUpdateOrbitData ( )
```

Forces the update of internal orbit data from current world positions of body, attractor settings and velocityHandle.

This method must be called after any manual changing of body position, velocity handler position or attractor settings. It will update internal OrbitData state from view state. Here is the call graph for this function:

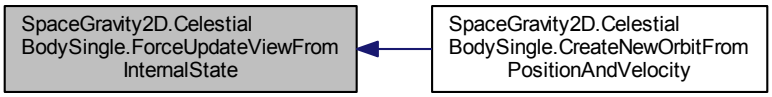


6.3.2.3 ForceUpdateViewFromInternalState()

```
void SpaceGravity2D.CelestialBodySingle.ForceUpdateViewFromInternalState ( )
```

Forces the update of body position, and velocity handler from OrbitData. Call this method after any direct changing of OrbitData.

Here is the caller graph for this function:

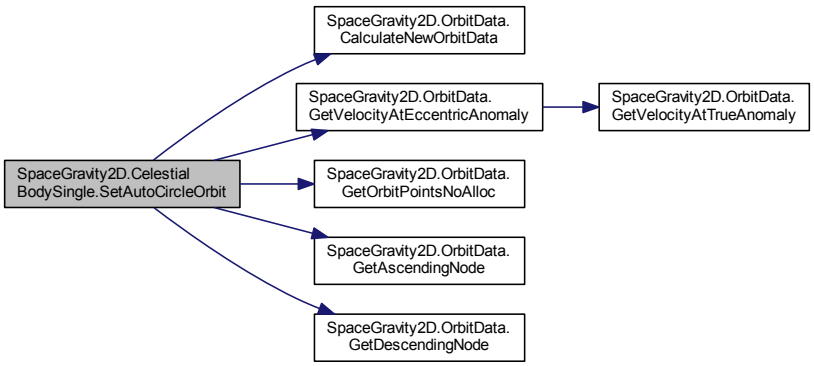


6.3.2.4 SetAutoCircleOrbit()

```
void SpaceGravity2D.CelestialBodySingle.SetAutoCircleOrbit ( )
```

Change orbit velocity vector to match circular orbit.

Here is the call graph for this function:



6.3.3 Member Data Documentation

6.3.3.1 AttractorMass

```
float SpaceGravity2D.CelestialBodySingle.AttractorMass = 1000
```

Attractor's mass. Should be bigger than 1.

6.3.3.2 AttractorObjectRef

```
Transform SpaceGravity2D.CelestialBodySingle.AttractorObjectRef
```

Reference to attractor transform.

6.3.3.3 GravitationalConstant

```
float SpaceGravity2D.CelestialBodySingle.GravitationalConstant = 0.001f
```

Gravitational constant.

F = G ∗ (m1 ∗ m2 / distance)

6.3.3.4 LineRendererRef

```
LineRenderer SpaceGravity2D.CelestialBodySingle.LineRendererRef
```

Reference to orbit linerenderer. Required only if orbit display is used.

6.3.3.5 LockOrbitEditing

```
bool SpaceGravity2D.CelestialBodySingle.LockOrbitEditing = false
```

Disable continious editing orbit in update loop, if you don't need it.

6.3.3.6 MaxOrbitWorldUnitsDistance

```
float SpaceGravity2D.CelestialBodySingle.MaxOrbitWorldUnitsDistance = 100f
```

Max distance for display orbit in world units.

6.3.3.7 OrbitData

```
OrbitData SpaceGravity2D.CelestialBodySingle.OrbitData = new OrbitData()
```

The orbit data. Internal state of orbit.

6.3.3.8 OrbitPointsCount

```
int SpaceGravity2D.CelestialBodySingle.OrbitPointsCount = 50
```

Max display orbit points count. More points - better precision.

6.3.3.9 TimeScale

```
float SpaceGravity2D.CelestialBodySingle.TimeScale = 1f
```

Motion total speed setting.

6.3.3.10 VelocityHandleRef

```
Transform SpaceGravity2D.CelestialBodySingle.VelocityHandleRef
```

Reference to velocity handle object. Assign object and use it as velocity control handle in scene view.

6.3.3.11 VelocityMlt

```
float SpaceGravity2D.CelestialBodySingle.VelocityMlt = 1f
```

Multiplier for velocity;

6.3.4 Property Documentation

6.3.4.1 attractorMass

```
float SpaceGravity2D.CelestialBodySingle.attractorMass  [get]
```

Attractor's mass. Should be bigger than 1.

6.3.4.2 attractorObject

```
Transform SpaceGravity2D.CelestialBodySingle.attractorObject  [get]
```

Reference to attractor transform.

6.3.4.3 G

```
float SpaceGravity2D.CelestialBodySingle.G  [get]
```

Gravitational constant.

6.3.4.4 linerend

```
LineRenderer SpaceGravity2D.CelestialBodySingle.linerend  [get]
```

Reference to orbit linerenderer.

6.3.4.5 maxDistForHyperbolicCase

```
float SpaceGravity2D.CelestialBodySingle.maxDistForHyperbolicCase [get]
```

Max distance for display orbit.

6.3.4.6 orbitData

```
OrbitData SpaceGravity2D.CelestialBodySingle.orbitData [get]
```

The orbit data. Internal state of orbit.

6.3.4.7 orbitPointsCount

```
int SpaceGravity2D.CelestialBodySingle.orbitPointsCount [get]
```

6.3.4.8 velocityHandle

```
Transform SpaceGravity2D.CelestialBodySingle.velocityHandle [get]
```

Reference to velocity handle object. Assign object and use it as velocity control handle in scene view.

6.3.4.9 velocityMlt

```
float SpaceGravity2D.CelestialBodySingle.velocityMlt [get]
```

Multiplier for velocity;

The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/CelestialBodySingle.cs

6.4 SpaceGravity2D.OrbitData Class Reference

Orbit data container. Also contains methods for altering and updating orbit state.

Public Member Functions

- void CalculateNewOrbitData ()

  Calculates the full state of orbit from current body position, attractor position, attractor mass, velocity, and grav↩ Constant.
- Vector3d GetVelocityAtEccentricAnomaly (double eccentricAnomaly)

  Gets the velocity vector value at eccentric anomaly.
- Vector3d GetVelocityAtTrueAnomaly (double trueAnomaly)

  Gets the velocity value at true anomaly.
- Vector3d GetCentralPositionAtTrueAnomaly (double trueAnomaly)

  Gets the central position at true anomaly.
- Vector3d GetCentralPositionAtEccentricAnomaly (double eccentricAnomaly)

  Gets the central position at eccentric anomaly.
- Vector3d GetFocalPositionAtEccentricAnomaly (double eccentricAnomaly)

  Gets the focal position at eccentric anomaly.
- Vector3d GetFocalPositionAtTrueAnomaly (double trueAnomaly)

  Gets the focal position at true anomaly.
- Vector3d GetCentralPosition ()

  Gets the central position.
- Vector3d [ ] GetOrbitPoints (int pointsCount=50, double maxDistance=1000d)

  Get orbit curve points if current orbit state is valid.
- Vector3d [ ] GetOrbitPoints (int pointsCount, Vector3d origin, double maxDistance=1000d)

  Get orbit curve points if current orbit state is valid.
- void GetOrbitPointsNoAlloc (ref Vector3d[ ] points, int pointsCount=50, double maxDistance=1000d)

  Get orbit curve points without array allocation, if current orbit state is valid.
- void GetOrbitPointsNoAlloc (ref Vector3d[ ] points, int pointsCount, Vector3d origin, double max↩ Distance=1000d)

  Get orbit curve points without array allocation, if current orbit state is valid.
- Vector3 [ ] GetOrbitPoints (int pointsCount=50, float maxDistance=1000f)

  Get orbit curve points if current orbit state is valid.
- Vector3 [ ] GetOrbitPoints (int pointsCount, Vector3 origin, float maxDistance=1000f)

  Get orbit curve points if current orbit state is valid.
- void GetOrbitPointsNoAlloc (ref Vector3[ ] points, int pointsCount=50, float maxDistance=1000f)

  Get orbit curve points without array allocation, if current orbit state is valid.
- void GetOrbitPointsNoAlloc (ref Vector3[ ] points, int pointsCount, Vector3 origin, float maxDistance=1000f)

  Get orbit curve points without array allocation, if current orbit state is valid.
- bool GetAscendingNode (out Vector3 asc)

  Gets the ascending node of orbit.
- bool GetAscendingNode (out Vector3d asc)

  Gets the ascending node of orbit.
- bool GetDescendingNode (out Vector3 desc)

  Gets the descending node of orbit.
- bool GetDescendingNode (out Vector3d desc)

  Gets the descending node of orbit.
- void UpdateOrbitDataByTime (double deltaTime)

  Updates the kepler orbit state by defined deltatime. Orbit main parameters will remains unchanged, but all anomalies will progress in time.
- void UpdateOrbitAnomaliesByTime (double deltaTime)

  Updates the value of orbital anomalies by defined deltatime.
- void SetPositionByCurrentAnomaly ()

  Updates position from eccentric anomaly state.

- void SetVelocityByCurrentAnomaly ()

  Sets velocity by current eccentric anomaly.
- void SetEccentricity (double e)

  Sets the eccentricity and updates all corresponding orbit state values.
- void SetMeanAnomaly (double m)

  Sets the mean anomaly and updates all other anomalies.
- void SetTrueAnomaly (double t)

  Sets the true anomaly and updates all other anomalies.
- void SetEccentricAnomaly (double e)

  Sets the eccentric anomaly and updates all other anomalies.
- void RotateOrbit (Quaternion rotation)

  Rotates the relative position and velocity by same quaternion.

Public Attributes

- double Epsilon = 1e-010

  Minimal floating point value.
- double GravitationalConstant = 0.001

  Gravitational force coeficient.
- Vector3d EclipticNormal = new Vector3d(0, 0, 1)

  Normal vecotr of ecliptic plane.
- Vector3d EclipticUp = new Vector3d(0, 1, 0)

  Perpendicular to EclipticNormal vector. Represents up direction on ecliptic plane.
- Vector3d Position

  Current position of the body in local orbit space.
- double AttractorDistance

  Distance to attractor.
- double AttractorMass

  Mass of the attractor.
- Vector3d Velocity

  Current velocity direction and magnitude.
- double SemiMinorAxis

  Magnitude of semi minor axis of the orbit's elliptic curve.
- double SemiMajorAxis

  Magnitude of semi minor axis of the orbit's elliptic curve.
- double FocalParameter

  Focal parameter of orbit's elliptic curve.
- double Eccentricity

  Eccentricity of orbit's elliptic curve.
- double EnergyTotal

  Total kinetic and potential energy of the orbit.
- double Period

  Period time of once cycle in seconds (if orbit is not hyperbolic).
- double TrueAnomaly

  True anomaly in radians.
- double MeanAnomaly

  Mean anomaly in radians.
- double EccentricAnomaly

  Eccentric anomaly in radians.
- double SquaresConstant

Square-constant parameter for orbit's elliptic curve.

- Vector3d Periapsis

  Periapsis point of the orbit.
- double PeriapsisDistance

  Distance to periapsis from the main focus point of orbit's elliptic curve.
- Vector3d Apoapsis

  Apoapsis point of the orbit. Not defined for hyperbolic orbits.
- double ApoapsisDistance

  Distance to apoapsis from the main focus point of orbit's elliptic curve.
- Vector3d CenterPoint

  Position of center of orbit's elliptic curve relative to main focus point.
- double OrbitCompressionRatio

  Compression parameter of orbit's elliptic curve.
- Vector3d OrbitNormal

  Perpendicular vector to orbit's plane.
- Vector3d SemiMinorAxisBasis

  Basis vector (direction) of Semi Minor Axis of the orbit.
- Vector3d SemiMajorAxisBasis

  Basis vector (direction) of Semi Major Axis of the orbit.
- double Inclination

  The orbit inclination in radians relative to ecliptic plane.
- double OrbitNormalDotEclipticNormal

  if $>$ 0, then orbit motion is clockwise.
- bool IsDirty = false

  Was orbit state changed without recalculating.

Properties

- bool IsValidOrbit `[get]`

  Is orbit state valid and error-free.

6.4.1 Detailed Description

Orbit data container. Also contains methods for altering and updating orbit state.

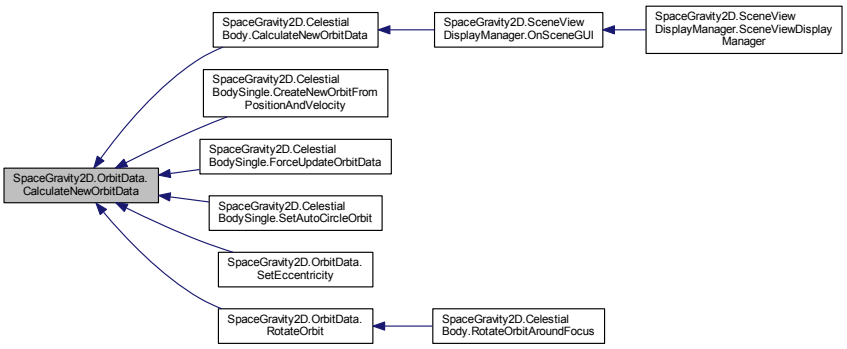
6.4.2 Member Function Documentation

6.4.2.1 CalculateNewOrbitData()

```
void SpaceGravity2D.OrbitData.CalculateNewOrbitData ( )
```

Calculates the full state of orbit from current body position, attractor position, attractor mass, velocity, and grav↩ Constant.

Here is the caller graph for this function:



6.4.2.2 GetAscendingNode() [1/2]

```
bool SpaceGravity2D.OrbitData.GetAscendingNode (
            out Vector3 asc )
```

Gets the ascending node of orbit.

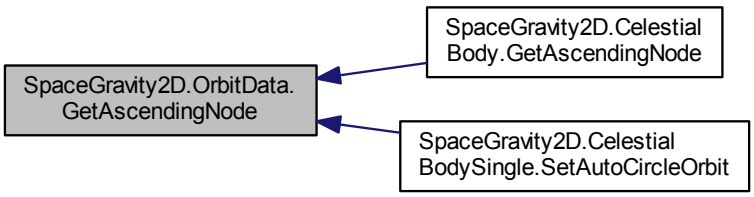
Parameters

asc	The asc.

Returns

`true` if ascending node exists, otherwise `false`

Here is the caller graph for this function:



6.4.2.3 GetAscendingNode() [2/2]

```
bool SpaceGravity2D.OrbitData.GetAscendingNode (
            out Vector3d asc )
```

Gets the ascending node of orbit.

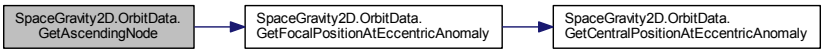
Parameters

asc	The asc.

Returns

`true` if ascending node exists, otherwise `false`

Here is the call graph for this function:



6.4.2.4 GetCentralPosition()

```
Vector3d SpaceGravity2D.OrbitData.GetCentralPosition ( )
```

Gets the central position.

Returns

Position relative to orbit center.

Note: central position is not same as focal position.

6.4.2.5 GetCentralPositionAtEccentricAnomaly()

```
Vector3d SpaceGravity2D.OrbitData.GetCentralPositionAtEccentricAnomaly (
            double eccentricAnomaly )
```

Gets the central position at eccentric anomaly.

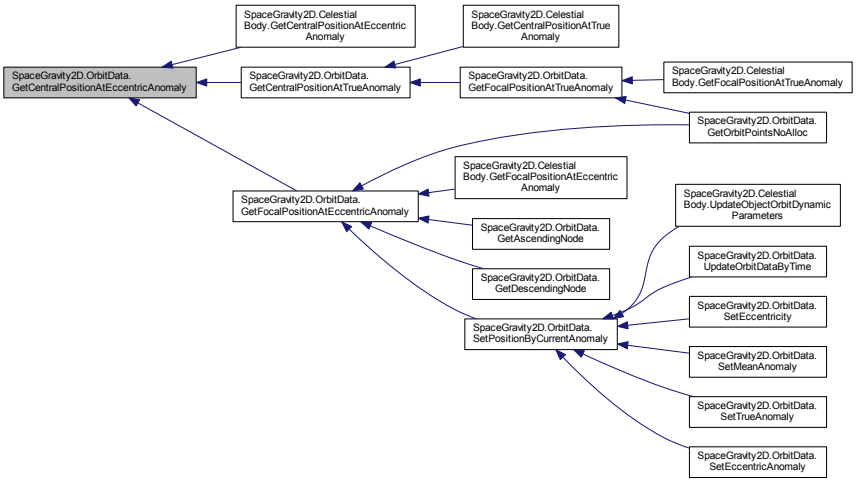
Parameters

eccentricAnomaly	The eccentric anomaly.

Returns

Position relative to orbit center.

Note: central position is not same as focal position. Here is the caller graph for this function:



6.4.2.6 GetCentralPositionAtTrueAnomaly()

```
Vector3d SpaceGravity2D.OrbitData.GetCentralPositionAtTrueAnomaly (
            double trueAnomaly )
```

Gets the central position at true anomaly.

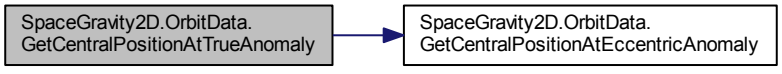
Parameters

trueAnomaly	The true anomaly.

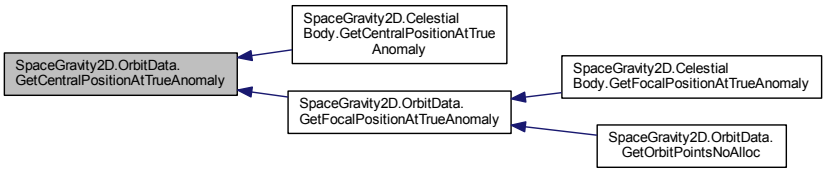
Returns

Position relative to orbit center.

Note: central position is not same as focal position. Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.7 GetDescendingNode() [1/2]

```
bool SpaceGravity2D.OrbitData.GetDescendingNode (
            out Vector3 desc )
```

Gets the descending node of orbit.

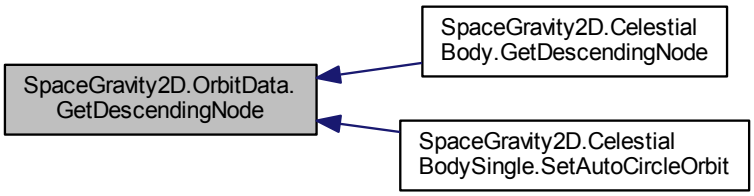
Parameters

desc	The desc.

Returns

`true` if descending node exists, otherwise `false`

Here is the caller graph for this function:



6.4.2.8 GetDescendingNode() [2/2]

```
bool SpaceGravity2D.OrbitData.GetDescendingNode (
            out Vector3d desc )
```

Gets the descending node of orbit.

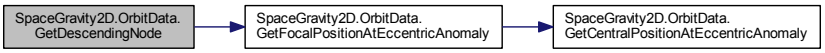
Parameters

desc	The desc.

Returns

`true` if descending node exists, otherwise `false`

Here is the call graph for this function:



6.4.2.9 GetFocalPositionAtEccentricAnomaly()

```
Vector3d SpaceGravity2D.OrbitData.GetFocalPositionAtEccentricAnomaly (
            double eccentricAnomaly )
```

Gets the focal position at eccentric anomaly.

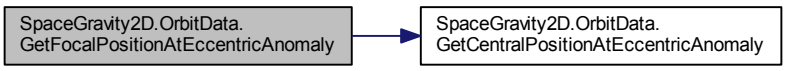
Parameters

eccentricAnomaly	The eccentric anomaly.

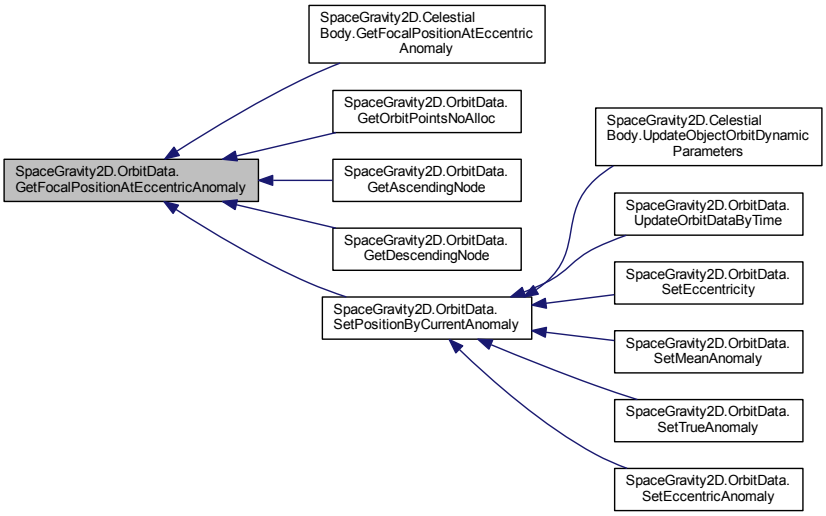
Returns

Position relative to attractor (focus).

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.10 GetFocalPositionAtTrueAnomaly()

```
Vector3d SpaceGravity2D.OrbitData.GetFocalPositionAtTrueAnomaly (
            double trueAnomaly )
```

Gets the focal position at true anomaly.

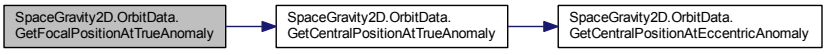
Parameters

trueAnomaly	The true anomaly.

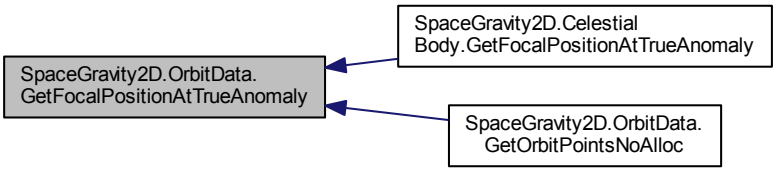
Returns

Position relative to attractor (focus).

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.11 GetOrbitPoints() [1/4]

```
Vector3d [] SpaceGravity2D.OrbitData.GetOrbitPoints (
            int pointsCount = 50,
            double maxDistance = 1000d )
```

Get orbit curve points if current orbit state is valid.

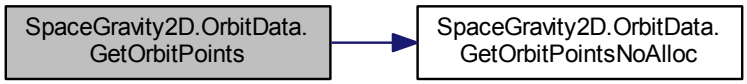
Parameters

pointsCount	Max points count in curve.
maxDistance	Max distance for points in curve.

Returns

Orbit curve points array.

Here is the call graph for this function:



6.4.2.12 GetOrbitPoints() [2/4]

```
Vector3d [] SpaceGravity2D.OrbitData.GetOrbitPoints (
            int pointsCount,
            Vector3d origin,
            double maxDistance = 1000d )
```

Get orbit curve points if current orbit state is valid.

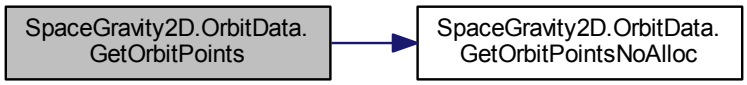
Parameters

pointsCount	Max points count in curve.
origin	World position of attractor (focus of orbit).
maxDistance	Max distance for points in curve.

Returns

Orbit curve points array.

Here is the call graph for this function:



6.4.2.13 GetOrbitPoints() [3/4]

```
Vector3 [] SpaceGravity2D.OrbitData.GetOrbitPoints (
            int pointsCount = 50,
            float maxDistance = 1000f )
```

Get orbit curve points if current orbit state is valid.

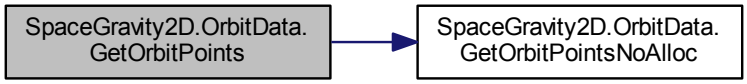
Parameters

pointsCount	Max orbit curve points count.
maxDistance	Max distance for orbit curve points.

Returns

Orbit curve points array.

Here is the call graph for this function:



6.4.2.14 GetOrbitPoints() [4/4]

```
Vector3 [] SpaceGravity2D.OrbitData.GetOrbitPoints (
            int pointsCount,
            Vector3 origin,
            float maxDistance = 1000f )
```

Get orbit curve points if current orbit state is valid.

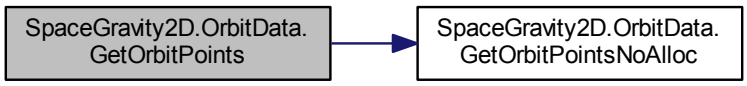
Parameters

pointsCount	Max orbit curve points count.
origin	World position of attractor (focus of orbit).
maxDistance	Max distance for orbit curve points.

Returns

Orbit curve points array.

Here is the call graph for this function:



6.4.2.15 GetOrbitPointsNoAlloc() [1/4]

```
void SpaceGravity2D.OrbitData.GetOrbitPointsNoAlloc (
            ref Vector3d [] points,
            int pointsCount = 50,
            double maxDistance = 1000d )
```

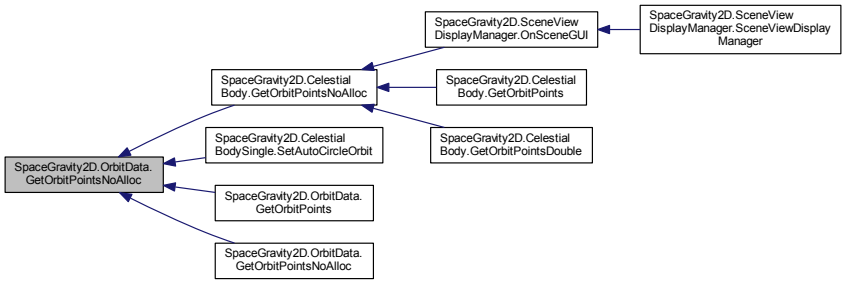
Get orbit curve points without array allocation, if current orbit state is valid.

Note: array allocation may sometimes occur, if specified array is null or lenght is not equal to target points count.

Parameters

points	Resulting orbit curve array.
pointsCount	Max orbit curve points count.
maxDistance	Max distance for orbit curve points.

Here is the caller graph for this function:



6.4.2.16 GetOrbitPointsNoAlloc() [2/4]

```
void SpaceGravity2D.OrbitData.GetOrbitPointsNoAlloc (
            ref Vector3d [] points,
            int pointsCount,
            Vector3d origin,
            double maxDistance = 1000d )
```

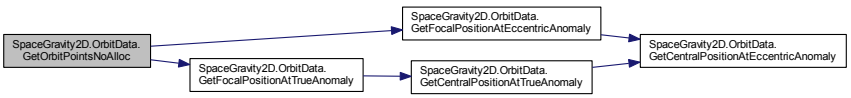
Get orbit curve points without array allocation, if current orbit state is valid.

Note: array allocation may sometimes occur, if specified array is null or lenght is not equal to target points count.

Parameters

points	Resulting orbit curve array.
pointsCount	Max orbit curve points count.
origin	World position of attractor (focus of orbit).
maxDistance	Max distance for orbit curve points.

Here is the call graph for this function:



6.4.2.17 GetOrbitPointsNoAlloc() [3/4]

```
void SpaceGravity2D.OrbitData.GetOrbitPointsNoAlloc (
            ref Vector3 [] points,
            int pointsCount = 50,
            float maxDistance = 1000f )
```

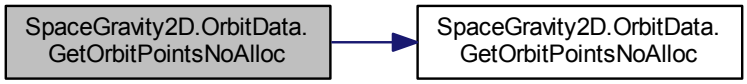
Get orbit curve points without array allocation, if current orbit state is valid.

Note: array allocation may sometimes occur, if specified array is null or lenght is not equal to target points count.

Parameters

points	Resulting orbit curve array.
pointsCount	Max orbit curve points count.
maxDistance	Max distance for orbit curve points.

Here is the call graph for this function:



6.4.2.18 GetOrbitPointsNoAlloc() [4/4]

```
void SpaceGravity2D.OrbitData.GetOrbitPointsNoAlloc (
            ref Vector3 [] points,
            int pointsCount,
            Vector3 origin,
            float maxDistance = 1000f )
```

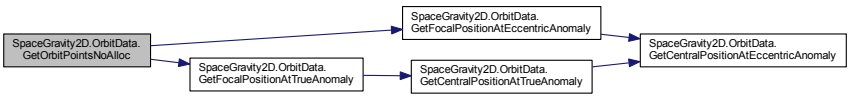
Get orbit curve points without array allocation, if current orbit state is valid.

Note: array allocation may sometimes occur, if specified array is null or lenght is not equal to target points count.

Parameters

points	Resulting orbit curve array.
pointsCount	Max orbit curve points count.
origin	World position of attractor (focus of orbit).
maxDistance	Max distance for orbit curve points.

Here is the call graph for this function:



6.4.2.19 GetVelocityAtEccentricAnomaly()

```
Vector3d SpaceGravity2D.OrbitData.GetVelocityAtEccentricAnomaly (
            double eccentricAnomaly )
```

Gets the velocity vector value at eccentric anomaly.

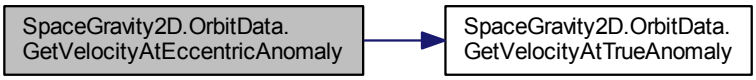
Parameters

eccentricAnomaly	The eccentric anomaly.

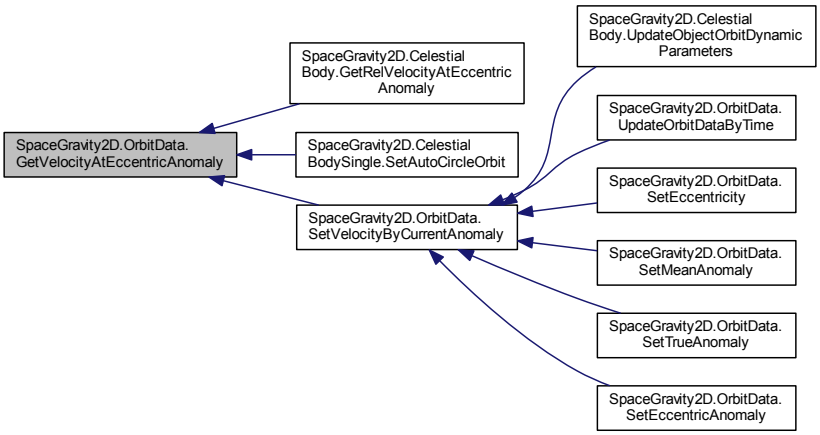
Returns

Velocity vector.

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.20 GetVelocityAtTrueAnomaly()

```
Vector3d SpaceGravity2D.OrbitData.GetVelocityAtTrueAnomaly (
            double trueAnomaly )
```

Gets the velocity value at true anomaly.

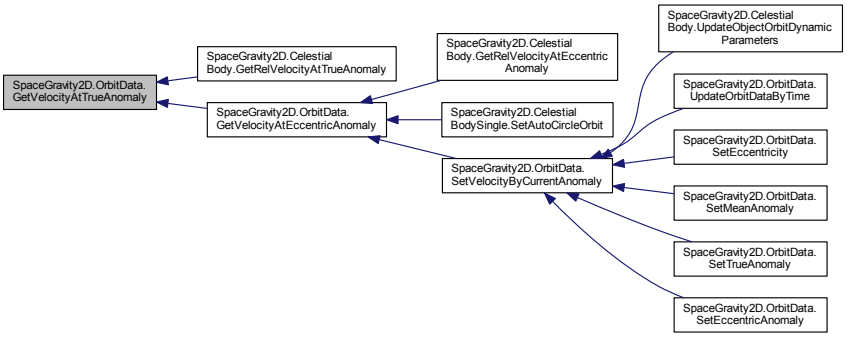
Parameters

trueAnomaly	The true anomaly.

Returns

Velocity vector.

Here is the caller graph for this function:



6.4.2.21 RotateOrbit()

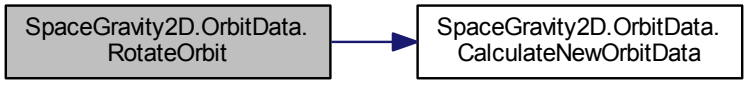
```
void SpaceGravity2D.OrbitData.RotateOrbit (
            Quaternion rotation )
```

Rotates the relative position and velocity by same quaternion.

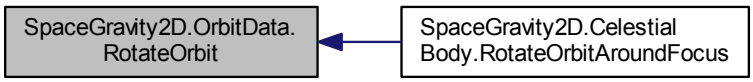
Parameters

rotation	The rotation.

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.22 SetEccentricAnomaly()

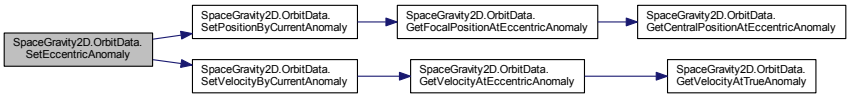
```
void SpaceGravity2D.OrbitData.SetEccentricAnomaly (
            double e )
```

Sets the eccentric anomaly and updates all other anomalies.

Parameters

e	The e.

Here is the call graph for this function:



6.4.2.23 SetEccentricity()

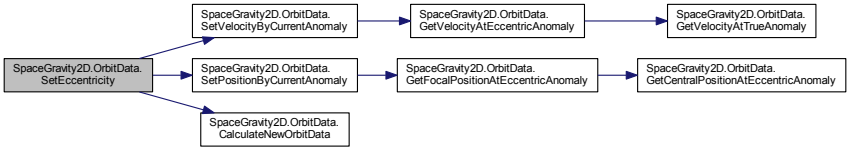
```
void SpaceGravity2D.OrbitData.SetEccentricity (
            double e )
```

Sets the eccentricity and updates all corresponding orbit state values.

Parameters

e	The new eccentricity value.

Mean anomaly will try to preserve. Here is the call graph for this function:



6.4.2.24 SetMeanAnomaly()

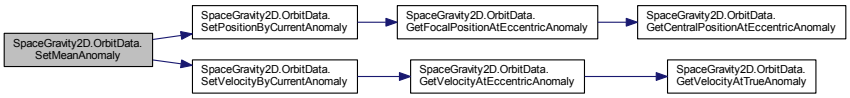
```
void SpaceGravity2D.OrbitData.SetMeanAnomaly (
            double m )
```

Sets the mean anomaly and updates all other anomalies.

Parameters

m	The m.

Here is the call graph for this function:

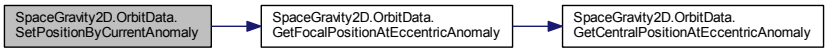


6.4.2.25 SetPositionByCurrentAnomaly()

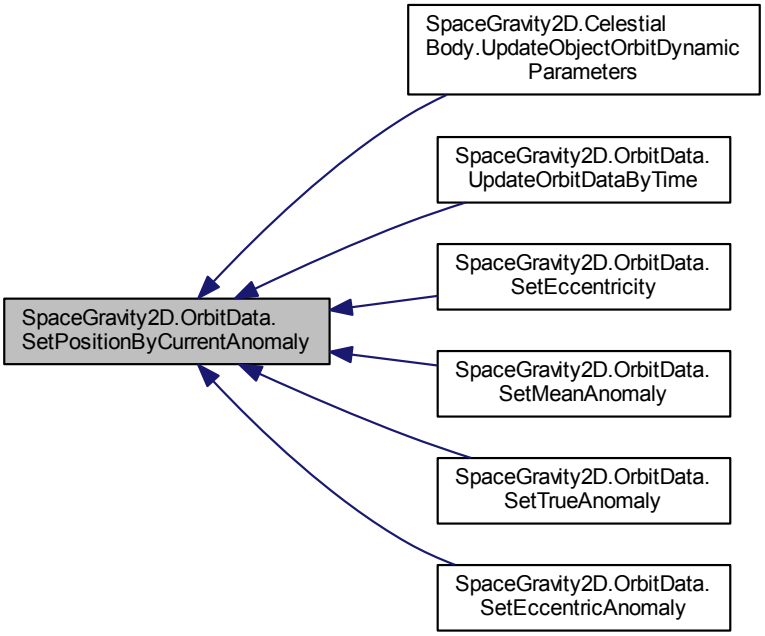
```
void SpaceGravity2D.OrbitData.SetPositionByCurrentAnomaly ( )
```

Updates position from eccentric anomaly state.

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.26 SetTrueAnomaly()

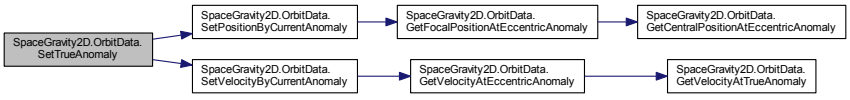
```
void SpaceGravity2D.OrbitData.SetTrueAnomaly (
            double t )
```

Sets the true anomaly and updates all other anomalies.

Parameters

t	The t.

Here is the call graph for this function:

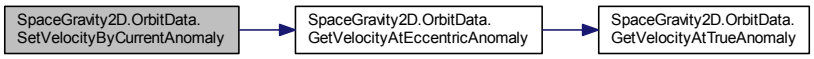


6.4.2.27 SetVelocityByCurrentAnomaly()

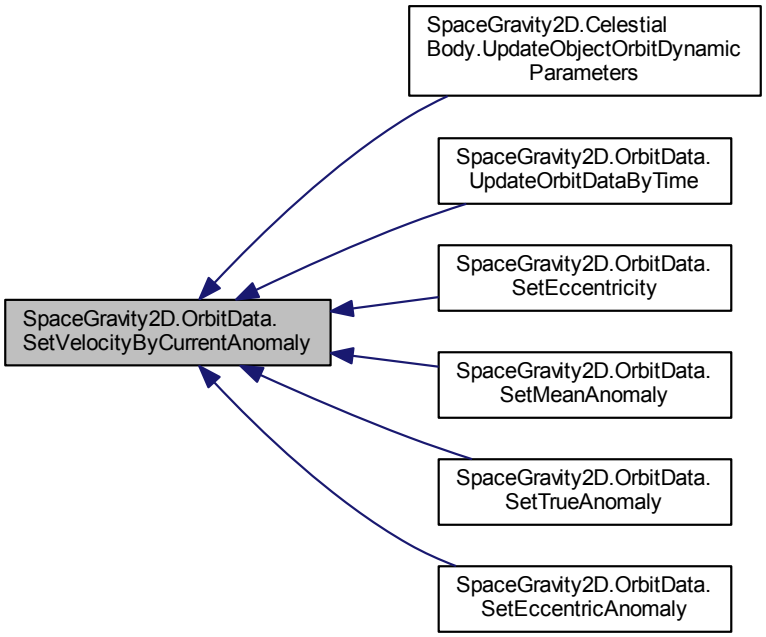
```
void SpaceGravity2D.OrbitData.SetVelocityByCurrentAnomaly ( )
```

Sets velocity by current eccentric anomaly.

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2.28 UpdateOrbitAnomaliesByTime()

```
void SpaceGravity2D.OrbitData.UpdateOrbitAnomaliesByTime (
            double deltaTime )
```

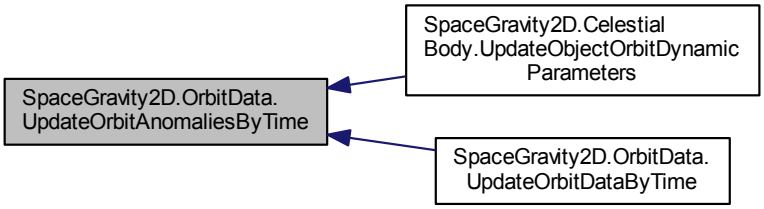
Updates the value of orbital anomalies by defined deltatime.

Parameters

deltaTime	The delta time.

Only anomalies values will be changed. Position and velocity states needs to be updated too after this method call.

Here is the caller graph for this function:



6.4.2.29 UpdateOrbitDataByTime()

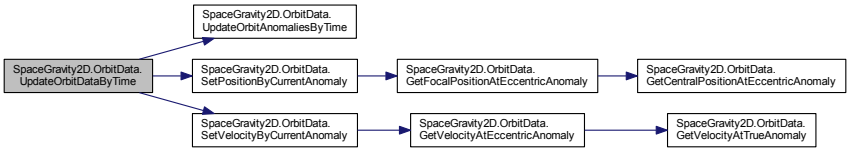
```
void SpaceGravity2D.OrbitData.UpdateOrbitDataByTime (
            double deltaTime )
```

Updates the kepler orbit state by defined deltatime. Orbit main parameters will remains unchanged, but all anomalies will progress in time.

Parameters

deltaTime	The delta time.

Here is the call graph for this function:



6.4.3 Member Data Documentation

6.4.3.1 Apoapsis

```
Vector3d SpaceGravity2D.OrbitData.Apoapsis
```

Apoapsis point of the orbit. Not defined for hyperbolic orbits.

6.4.3.2 ApoapsisDistance

```
double SpaceGravity2D.OrbitData.ApoapsisDistance
```

Distance to apoapsis from the main focus point of orbit's elliptic curve.

6.4.3.3 AttractorDistance

```
double SpaceGravity2D.OrbitData.AttractorDistance
```

Distance to attractor.

6.4.3.4 AttractorMass

```
double SpaceGravity2D.OrbitData.AttractorMass
```

Mass of the attractor.

6.4.3.5 CenterPoint

```
Vector3d SpaceGravity2D.OrbitData.CenterPoint
```

Position of center of orbit's elliptic curve relative to main focus point.

6.4.3.6 EccentricAnomaly

```
double SpaceGravity2D.OrbitData.EccentricAnomaly
```

Eccentric anomaly in radians.

6.4.3.7 Eccentricity

```
double SpaceGravity2D.OrbitData.Eccentricity
```

Eccentricity of orbit's elliptic curve.

6.4.3.8 EclipticNormal

```
Vector3d SpaceGravity2D.OrbitData.EclipticNormal = new Vector3d(0, 0, 1)
```

Normal vecotr of ecliptic plane.

Ecliptic plane is used for decoration, or when orbit is limited to two dimensions.

6.4.3.9 EclipticUp

```
Vector3d SpaceGravity2D.OrbitData.EclipticUp = new Vector3d(0, 1, 0)
```

Perpendicular to EclipticNormal vector. Represents up direction on ecliptic plane.

6.4.3.10 EnergyTotal

```
double SpaceGravity2D.OrbitData.EnergyTotal
```

Total kinetic and potential energy of the orbit.

6.4.3.11 Epsilon

```
double SpaceGravity2D.OrbitData.Epsilon = 1e-010
```

Minimal floating point value.

6.4.3.12 FocalParameter

```
double SpaceGravity2D.OrbitData.FocalParameter
```

Focal parameter of orbit's elliptic curve.

6.4.3.13 GravitationalConstant

```
double SpaceGravity2D.OrbitData.GravitationalConstant = 0.001
```

Gravitational force coeficient.

6.4.3.14 Inclination

```
double SpaceGravity2D.OrbitData.Inclination
```

The orbit inclination in radians relative to ecliptic plane.

6.4.3.15 IsDirty

```
bool SpaceGravity2D.OrbitData.IsDirty = false
```

Was orbit state changed without recalculating.

6.4.3.16 MeanAnomaly

```
double SpaceGravity2D.OrbitData.MeanAnomaly
```

Mean anomaly in radians.

6.4.3.17 OrbitCompressionRatio

```
double SpaceGravity2D.OrbitData.OrbitCompressionRatio
```

Compression parameter of orbit's elliptic curve.

6.4.3.18 OrbitNormal

```
Vector3d SpaceGravity2D.OrbitData.OrbitNormal
```

Perpendicular vector to orbit's plane.

6.4.3.19 OrbitNormalDotEclipticNormal

```
double SpaceGravity2D.OrbitData.OrbitNormalDotEclipticNormal
```

if > 0, then orbit motion is clockwise.

6.4.3.20 Periapsis

```
Vector3d SpaceGravity2D.OrbitData.Periapsis
```

Periapsis point of the orbit.

6.4.3.21 PeriapsisDistance

```
double SpaceGravity2D.OrbitData.PeriapsisDistance
```

Distance to periapsis from the main focus point of orbit's elliptic curve.

6.4.3.22 Period

```
double SpaceGravity2D.OrbitData.Period
```

Period time of once cycle in seconds (if orbit is not hyperbolic).

6.4.3.23 Position

```
Vector3d SpaceGravity2D.OrbitData.Position
```

Current position of the body in local orbit space.

6.4.3.24 SemiMajorAxis

```
double SpaceGravity2D.OrbitData.SemiMajorAxis
```

Magnitude of semi minor axis of the orbit's elliptic curve.

6.4.3.25 SemiMajorAxisBasis

```
Vector3d SpaceGravity2D.OrbitData.SemiMajorAxisBasis
```

Basis vector (direction) of Semi Major Axis of the orbit.

6.4.3.26 SemiMinorAxis

```
double SpaceGravity2D.OrbitData.SemiMinorAxis
```

Magnitude of semi minor axis of the orbit's elliptic curve.

6.4.3.27 SemiMinorAxisBasis

```
Vector3d SpaceGravity2D.OrbitData.SemiMinorAxisBasis
```

Basis vector (direction) of Semi Minor Axis of the orbit.

6.4.3.28 SquaresConstant

```
double SpaceGravity2D.OrbitData.SquaresConstant
```

Square-constant parameter for orbit's elliptic curve.

6.4.3.29 TrueAnomaly

```
double SpaceGravity2D.OrbitData.TrueAnomaly
```

True anomaly in radians.

6.4.3.30 Velocity

```
Vector3d SpaceGravity2D.OrbitData.Velocity
```

Current velocity direction and magnitude.

6.4.4 Property Documentation

6.4.4.1 IsValidOrbit

```
bool SpaceGravity2D.OrbitData.IsValidOrbit  [get]
```

Is orbit state valid and error-free.

`true` if this instance is valid orbit; otherwise, `false`.

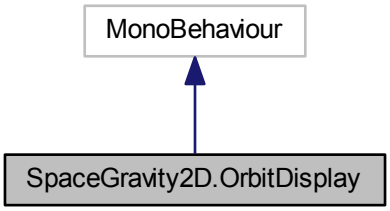
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/OrbitData.cs

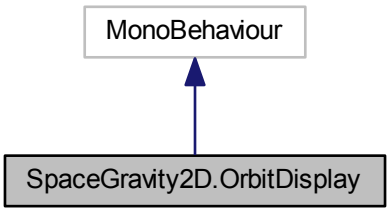
6.5 SpaceGravity2D.OrbitDisplay Class Reference

Component for displaying current orbit of CelestialBody. CelestialBody should be attached to same Game Object.

Inheritance diagram for SpaceGravity2D.OrbitDisplay:



Collaboration diagram for SpaceGravity2D.OrbitDisplay:



Public Member Functions

- void HideOrbit ()

Public Attributes

- Material OrbitLineMaterial

  Material for LineRenderer.
- float Width = 0.1f

  Line width.
- int OrbitPointsCount = 50

  Path points count. More points - better precision.
- float MaxOrbitPointsDistance = 100

  Max distance for orbit display in world units.
- LineRenderer LineRenderer

  Reference to line renderer.

6.5.1 Detailed Description

Component for displaying current orbit of CelestialBody. CelestialBody should be attached to same Game Object.

6.5.2 Member Function Documentation

6.5.2.1 HideOrbit()

```
void SpaceGravity2D.OrbitDisplay.HideOrbit ( )
```

6.5.3 Member Data Documentation

6.5.3.1 LineRenderer

```
LineRenderer SpaceGravity2D.OrbitDisplay.LineRenderer
```

Reference to line renderer.

6.5.3.2 MaxOrbitPointsDistance

```
float SpaceGravity2D.OrbitDisplay.MaxOrbitPointsDistance = 100
```

Max distance for orbit display in world units.

6.5.3.3 OrbitLineMaterial

```
Material SpaceGravity2D.OrbitDisplay.OrbitLineMaterial
```

Material for LineRenderer.

6.5.3.4 OrbitPointsCount

```
int SpaceGravity2D.OrbitDisplay.OrbitPointsCount = 50
```

Path points count. More points - better precision.

6.5.3.5 Width

```
float SpaceGravity2D.OrbitDisplay.Width = 0.1f
```

Line width.

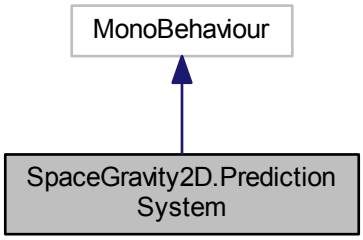
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/OrbitDisplay.cs

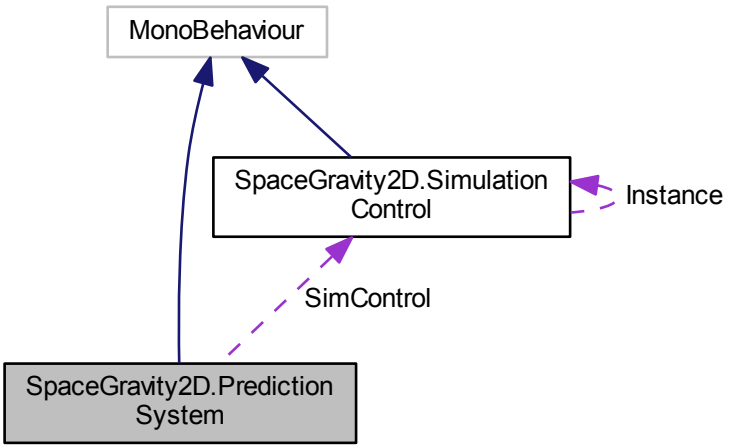
6.6 SpaceGravity2D.PredictionSystem Class Reference

Basic prediction orbits calculator (singleton). Simulates whole scene with Euler n-body algorythm PointCount steps into future, and displays resulting orbits with linerenderers.

Inheritance diagram for SpaceGravity2D.PredictionSystem:



Collaboration diagram for SpaceGravity2D.PredictionSystem:



Public Member Functions

- void HideAllOrbits ()

Public Attributes

- SimulationControl SimControl

  Reference to scene's simulation control.
- float CalcStep = 1f

  Calculation step precision. Lower value - better precision.
- int PointsCount = 50

  Determines how many steps will be calculated into future.
- Material LinesMaterial

  Global LineRenderer material.
- float LinesWidth = 0.05f

  Global LineRenderer width.

6.6.1 Detailed Description

Basic prediction orbits calculator (singleton). Simulates whole scene with Euler n-body algorythm PointCount steps into future, and displays resulting orbits with linerenderers.

6.6.2 Member Function Documentation

6.6.2.1 HideAllOrbits()

```
void SpaceGravity2D.PredictionSystem.HideAllOrbits ( )
```

6.6.3 Member Data Documentation

6.6.3.1 CalcStep

```
float SpaceGravity2D.PredictionSystem.CalcStep = 1f
```

Calculation step precision. Lower value - better precision.

Lower value increase precision, but decrease prediction range. Higher value decrease precision, and prediction range is increasing proportionaly. Can be balanced by PointsCount setting (Higher CalcStep - less PointsCount and vise versa).

6.6.3.2 LinesMaterial

```
Material SpaceGravity2D.PredictionSystem.LinesMaterial
```

Global LineRenderer material.

6.6.3.3 LinesWidth

```
float SpaceGravity2D.PredictionSystem.LinesWidth = 0.05f
```

Global LineRenderer width.

6.6.3.4 PointsCount

```
int SpaceGravity2D.PredictionSystem.PointsCount = 50
```

Determines how many steps will be calculated into future.

If precision (The CalcStep) is low, pointsCount should be higher and vise versa.

6.6.3.5 SimControl

```
SimulationControl SpaceGravity2D.PredictionSystem.SimControl
```

Reference to scene's simulation control.

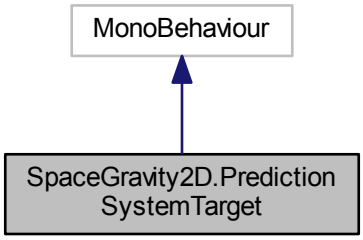
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/PredictionSystem.cs

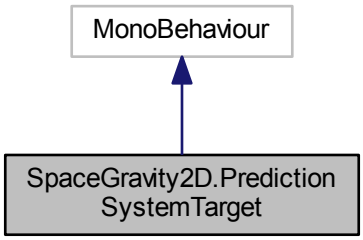
6.7 SpaceGravity2D.PredictionSystemTarget Class Reference

Component for celestial body object, which helps to control how PredictionSystem will display predicted motion path.

Inheritance diagram for SpaceGravity2D.PredictionSystemTarget:



Collaboration diagram for SpaceGravity2D.PredictionSystemTarget:



Public Attributes

- Material OrbitMaterial
- float OrbitWidth = 0.1f

6.7.1 Detailed Description

Component for celestial body object, which helps to control how PredictionSystem will display predicted motion path.

6.7.2 Member Data Documentation

6.7.2.1 OrbitMaterial

```
Material SpaceGravity2D.PredictionSystemTarget.OrbitMaterial
```

6.7.2.2 OrbitWidth

```
float SpaceGravity2D.PredictionSystemTarget.OrbitWidth = 0.1f
```

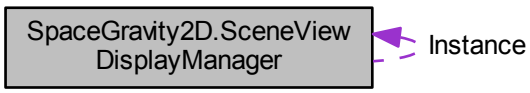
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/PredictionSystemTarget.cs

6.8 SpaceGravity2D.SceneViewDisplayManager Class Reference

Controller for HUD tools (orbits, buttons, labels display) in sceneview.

Collaboration diagram for SpaceGravity2D.SceneViewDisplayManager:



Public Member Functions

- SceneViewDisplayManager ()

  Initialize new instance of this type.
- void OnSceneGUI (SceneView sceneView)

  Draw all velocitiy vectors and orbits in scene window and process mouse dragging events.

Static Public Attributes

- static SceneViewDisplayManager Instance

  Singleton static reference.

Properties

- bool IsEclipticRotating `[get, set]`

  Get the current state of ecliptic rotation tool. Sets the new state of tool.
- bool IsOrbitRotating `[get, set]`

  Gets the state of orbit rotation tool. Sets the new state of the tool.
- bool IsVelocityRotating `[get, set]`

  Gets the current state of velocity rotation tool. Sets new state of the tool.

6.8.1 Detailed Description

Controller for HUD tools (orbits, buttons, labels display) in sceneview.

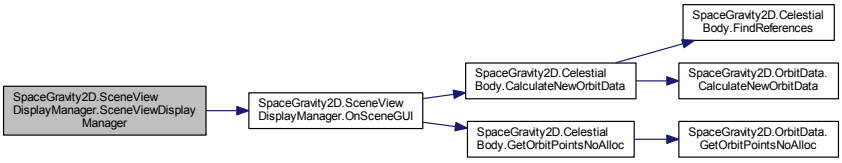
6.8.2 Constructor & Destructor Documentation

6.8.2.1 SceneViewDisplayManager()

```
SpaceGravity2D.SceneViewDisplayManager.SceneViewDisplayManager ( )
```

Initialize new instance of this type.

Here is the call graph for this function:



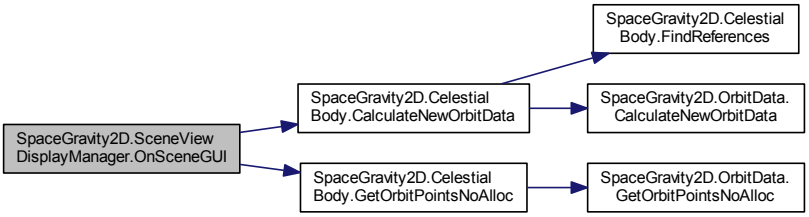
6.8.3 Member Function Documentation

6.8.3.1 OnSceneGUI()

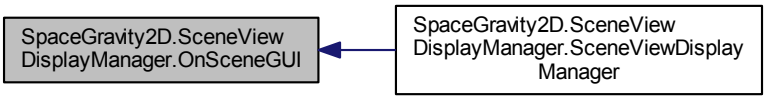
```
void SpaceGravity2D.SceneViewDisplayManager.OnSceneGUI (
            SceneView sceneView )
```

Draw all velocitiy vectors and orbits in scene window and process mouse dragging events.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4 Member Data Documentation

6.8.4.1 Instance

```
SceneViewDisplayManager SpaceGravity2D.SceneViewDisplayManager.Instance  [static]
```

Singleton static reference.

6.8.5 Property Documentation

6.8.5.1 IsEclipticRotating

```
bool SpaceGravity2D.SceneViewDisplayManager.IsEclipticRotating  [get], [set]
```

Get the current state of ecliptic rotation tool. Sets the new state of tool.

6.8.5.2 IsOrbitRotating

```
bool SpaceGravity2D.SceneViewDisplayManager.IsOrbitRotating  [get], [set]
```

Gets the state of orbit rotation tool. Sets the new state of the tool.

6.8.5.3 IsVelocityRotating

```
bool SpaceGravity2D.SceneViewDisplayManager.IsVelocityRotating  [get], [set]
```

Gets the current state of velocity rotation tool. Sets new state of the tool.

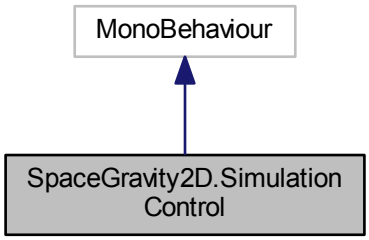
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Editor/SceneViewDisplayManager.cs

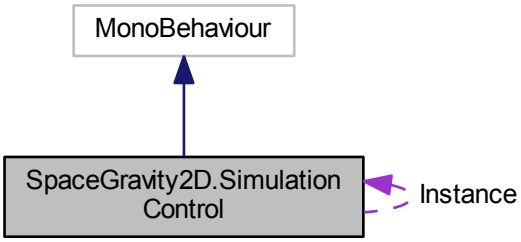
6.9 SpaceGravity2D.SimulationControl Class Reference

Main controller for gravitational motion on scene. Controls behaviour of celestial bodies, and holds global settings of gravitational simulation.

Inheritance diagram for SpaceGravity2D.SimulationControl:



Collaboration diagram for SpaceGravity2D.SimulationControl:



Public Types

- enum NBodyCalculationType { NBodyCalculationType.Euler = 0, NBodyCalculationType.Verlet, NBody↩ CalculationType.RungeKutta }

  List of available n-body calculation algorythms.

Public Member Functions

- SimulationControl ()
- void ProjectAllBodiesOnEcliptic ()
- CelestialBody FindMostProperAttractor (CelestialBody body)

  Find attractor, which have most gravitational influence at target body.
- CelestialBody FindBiggestAttractor ()

  Find attracter with biggest mass on scene.
- CelestialBody FindNearestAttractor (CelestialBody body)

  Find attractor with shortest distance to target body.
- void SetNearestAttractorForBody (CelestialBody body)

  Fast and simple way to find attractor; But note, that not always nearest attractor is most proper
- void SetMostProperAttractorForBody (CelestialBody body)

  Find attractor which has biggest gravitational influence on body comparing to others. If fail, null will be assigned. It can be used in realtime for implementing more precise transitions beetween spheres of influence, but performance cost is high
- void SetBiggestAttractorForBody (CelestialBody body)

  Assign biggest attractor on scene to target body.
- void ChangeAllVelocitiesByFactor (double multiplier)

  Used for changing gravitational parameter without breaking orbits.
- void ApplyGravConstToAllBodies ()

  Refresh gravitational constant of orbitData of all celestial bodies.
- void ApplyEclipticNormalsToAllBodies ()

  Set ecliptic value to OrbitData of all celestial bodies.

Public Attributes

- double MaxAttractionRange = double.PositiveInfinity

  Global constraint for gravitational attraction range.
- double MinAttractionRange = 0.1d

  Global constraint for gravitational attraction range.
- double TimeScale = 1d

  TimeScale of simulation process. May be dynamicaly changed, but very large values decreasing precision of calculations
- double MinAttractorMass = 100d

  Mass threshold for body to became attractor
- List< CelestialBody > Bodies = new List<CelestialBody>()

  References to all active celestial bodies on scene.
- NBodyCalculationType CalculationType = NBodyCalculationType.Verlet

  Current n-body simulation type.
- bool AffectedByGlobalTimescale

  Is simulation affected by Time.timescale value.
- bool KeepBodiesOnEclipticPlane

  Is bodies positions restricted to be in single plane.

Static Public Attributes

- static SimulationControl Instance

  Singletorn reference.

Properties

- double GravitationalConstant `[get, set]`

  Gets Gravitational constant value. Sets Gravitational constant for all active bodies.
- double GravitationalConstantProportional `[get, set]`

  Gets Gravitational constant value. Sets Gravitational constant for all active bodies and changes all velocities proportional for making orbits unchanged.
- double maxAttractionRange `[get, set]`

  Global constraint for gravitational attraction range.
- double minAttractionRange `[get, set]`

  Global constraint for gravitational attraction range.
- double timeScale `[get, set]`

  TimeScale of simulation process. May be dynamicaly changed, but very large values decreasing precision of calculations
- double minAttractorMass `[get, set]`

  Mass threshold for body to became attractor
- List< CelestialBody > bodies `[get]`
- static SimulationControl instance `[get]`

  Singletorn reference.
- NBodyCalculationType calculationType `[get, set]`

  Current n-body simulation type.
- bool affectedByGlobalTimescale `[get, set]`

  Is simulation affected by Time.timescale value.
- bool keepBodiesOnEclipticPlane `[get, set]`

  Is bodies positions restricted to be in single plane.
- Vector3d eclipticNormal `[get, set]`
- Vector3d EclipticNormal `[get, set]`

  Gets or sets ecliptic normal vector. Vector magnitude is always 1.
- Vector3d eclipticUp `[get, set]`

  Gets or sets ecliptic up direction vector. Vector magnitude is always 1.
- Vector3d EclipticUp `[get, set]`

  Gets or sets ecliptic up direction vector. Vector magnitude is always 1.

6.9.1 Detailed Description

Main controller for gravitational motion on scene. Controls behaviour of celestial bodies, and holds global settings of gravitational simulation.

6.9.2 Member Enumeration Documentation

6.9.2.1 NBodyCalculationType

`enum SpaceGravity2D.SimulationControl.NBodyCalculationType [strong]`

List of available n-body calculation algorythms.

Enumerator

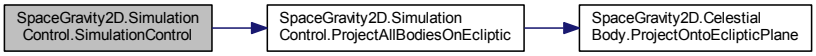
Euler	Fastest n-body algorythm.
Verlet	More stable n-body algorythm.
RungeKutta	Slowest and more precise n-body algorythm. Note - this algorythm may become very unstable if distance between attracting bodies becomes very close to value of position delta at current frame.

6.9.3 Constructor & Destructor Documentation

6.9.3.1 SimulationControl()

```
SpaceGravity2D.SimulationControl.SimulationControl ( )
```

Here is the call graph for this function:



6.9.4 Member Function Documentation

6.9.4.1 ApplyEclipticNormalsToAllBodies()

```
void SpaceGravity2D.SimulationControl.ApplyEclipticNormalsToAllBodies ( )
```

Set ecliptic value to OrbitData of all celestial bodies.

6.9.4.2 ApplyGravConstToAllBodies()

```
void SpaceGravity2D.SimulationControl.ApplyGravConstToAllBodies ( )
```

Refresh gravitational constant of orbitData of all celestial bodies.

6.9.4.3 ChangeAllVelocitiesByFactor()

```
void SpaceGravity2D.SimulationControl.ChangeAllVelocitiesByFactor (
            double multiplier )
```

Used for changing gravitational parameter without breaking orbits.

6.9.4.4 FindBiggestAttractor()

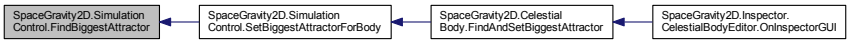
```
CelestialBody SpaceGravity2D.SimulationControl.FindBiggestAttractor ( )
```

Find attracter with biggest mass on scene.

Returns

Biggest attractor or null.

Here is the caller graph for this function:



6.9.4.5 FindMostProperAttractor()

```
CelestialBody SpaceGravity2D.SimulationControl.FindMostProperAttractor (
            CelestialBody body )
```

Find attractor, which have most gravitational influence at target body.

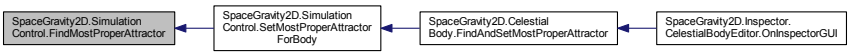
Parameters

body	Target body.

Returns

Most proper attractor or null.

Search logic: Calculate mutual perturbation for every pair of attractors in scene and select one, which attracts the body with biggest force and is least affected by others. Here is the caller graph for this function:



6.9.4.6 FindNearestAttractor()

```
CelestialBody SpaceGravity2D.SimulationControl.FindNearestAttractor (
            CelestialBody body )
```

Find attractor with shortest distance to target body.

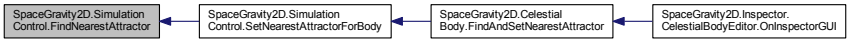
Parameters

body	Target body.

Returns

Nearest attractor or null.

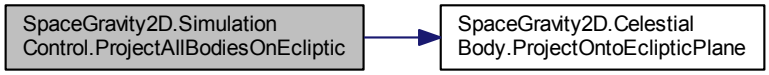
Here is the caller graph for this function:



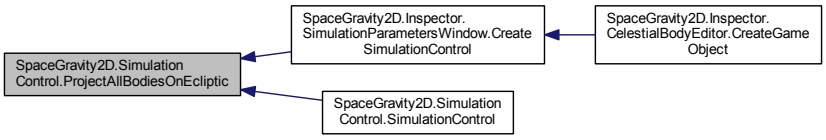
6.9.4.7 ProjectAllBodiesOnEcliptic()

```
void SpaceGravity2D.SimulationControl.ProjectAllBodiesOnEcliptic ( )
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.4.8 SetBiggestAttractorForBody()

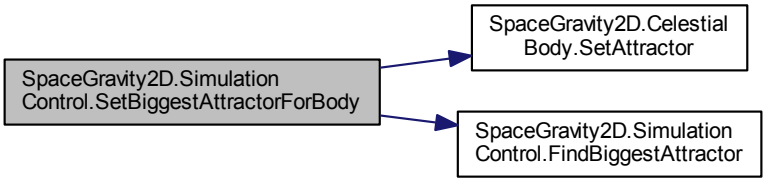
```
void SpaceGravity2D.SimulationControl.SetBiggestAttractorForBody (
            CelestialBody body )
```

Assign biggest attractor on scene to target body.

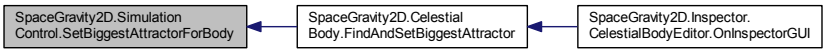
Parameters

body	

Here is the call graph for this function:



Here is the caller graph for this function:

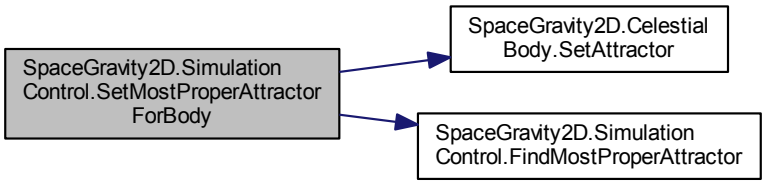


6.9.4.9 SetMostProperAttractorForBody()

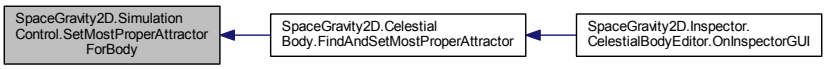
```
void SpaceGravity2D.SimulationControl.SetMostProperAttractorForBody (
            CelestialBody body )
```

Find attractor which has biggest gravitational influence on body comparing to others. If fail, null will be assigned. It can be used in realtime for implementing more precise transitions beetween spheres of influence, but performance cost is high

Here is the call graph for this function:



Here is the caller graph for this function:

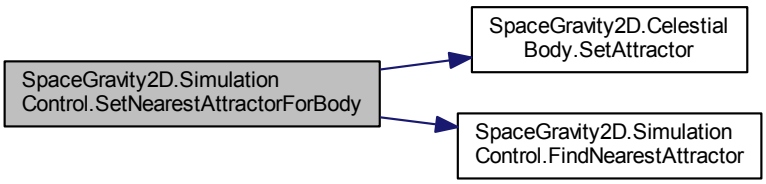


6.9.4.10 SetNearestAttractorForBody()

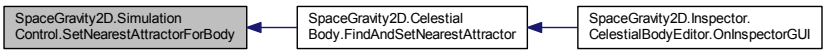
```
void SpaceGravity2D.SimulationControl.SetNearestAttractorForBody (
            CelestialBody body )
```

Fast and simple way to find attractor; But note, that not always nearest attractor is most proper

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.5 Member Data Documentation

6.9.5.1 AffectedByGlobalTimescale

```
bool SpaceGravity2D.SimulationControl.AffectedByGlobalTimescale
```

Is simulation affected by Time.timescale value.

6.9.5.2 Bodies

```
List<CelestialBody> SpaceGravity2D.SimulationControl.Bodies = new List<CelestialBody>()
```

References to all active celestial bodies on scene.

CelestialBody instance would register intself in this controller when it will be activated. This cache should not be serialized because it's contains runtime references only.

6.9.5.3 CalculationType

```
NBodyCalculationType SpaceGravity2D.SimulationControl.CalculationType = NBodyCalculation↩
Type.Verlet
```

Current n-body simulation type.

6.9.5.4 Instance

```
SimulationControl SpaceGravity2D.SimulationControl.Instance  [static]
```

Singletorn reference.

6.9.5.5 KeepBodiesOnEclipticPlane

```
bool SpaceGravity2D.SimulationControl.KeepBodiesOnEclipticPlane
```

Is bodies positions restricted to be in single plane.

May be used to make 2d world.

6.9.5.6 MaxAttractionRange

```
double SpaceGravity2D.SimulationControl.MaxAttractionRange = double.PositiveInfinity
```

Global constraint for gravitational attraction range.

6.9.5.7 MinAttractionRange

```
double SpaceGravity2D.SimulationControl.MinAttractionRange = 0.1d
```

Global constraint for gravitational attraction range.

It is better to set this value equal minimal body size. Not recommended to set 0 value, because infinity velocities will occur, when two bodies will approach each other too close.

6.9.5.8 MinAttractorMass

```
double SpaceGravity2D.SimulationControl.MinAttractorMass = 100d
```

Mass threshold for body to became attractor

6.9.5.9 TimeScale

```
double SpaceGravity2D.SimulationControl.TimeScale = 1d
```

TimeScale of simulation process. May be dynamicaly changed, but very large values decreasing precision of calculations

6.9.6 Property Documentation

6.9.6.1 affectedByGlobalTimescale

```
bool SpaceGravity2D.SimulationControl.affectedByGlobalTimescale [get], [set]
```

Is simulation affected by Time.timescale value.

6.9.6.2 bodies

```
List<CelestialBody> SpaceGravity2D.SimulationControl.bodies [get]
```

6.9.6.3 calculationType

```
NBodyCalculationType SpaceGravity2D.SimulationControl.calculationType [get], [set]
```

Current n-body simulation type.

6.9.6.4 eclipticNormal

```
Vector3d SpaceGravity2D.SimulationControl.eclipticNormal [get], [set]
```

6.9.6.5 EclipticNormal

```
Vector3d SpaceGravity2D.SimulationControl.EclipticNormal [get], [set]
```

Gets or sets ecliptic normal vector. Vector magnitude is always 1.

6.9.6.6 eclipticUp

```
Vector3d SpaceGravity2D.SimulationControl.eclipticUp [get], [set]
```

Gets or sets ecliptic up direction vector. Vector magnitude is always 1.

6.9.6.7 EclipticUp

```
Vector3d SpaceGravity2D.SimulationControl.EclipticUp  [get], [set]
```

Gets or sets ecliptic up direction vector. Vector magnitude is always 1.

6.9.6.8 GravitationalConstant

```
double SpaceGravity2D.SimulationControl.GravitationalConstant  [get], [set]
```

Gets Gravitational constant value. Sets Gravitational constant for all active bodies.

This property is usefull in runtime, when active celestial bodies have already fetched G constant from controller and it needs to be refreshed manually for all of them.

6.9.6.9 GravitationalConstantProportional

```
double SpaceGravity2D.SimulationControl.GravitationalConstantProportional  [get], [set]
```

Gets Gravitational constant value. Sets Gravitational constant for all active bodies and changes all velocities proportional for making orbits unchanged.

This property is usefull when need to change G const, but not need to change already customized orbits. Less G const value - slower motion of all objects is, and higher value - faster motion.

6.9.6.10 instance

```
SimulationControl SpaceGravity2D.SimulationControl.instance  [static], [get]
```

Singletorn reference.

6.9.6.11 keepBodiesOnEclipticPlane

```
bool SpaceGravity2D.SimulationControl.keepBodiesOnEclipticPlane  [get], [set]
```

Is bodies positions restricted to be in single plane.

6.9.6.12 maxAttractionRange

```
double SpaceGravity2D.SimulationControl.maxAttractionRange  [get], [set]
```

Global constraint for gravitational attraction range.

6.9.6.13 minAttractionRange

```
double SpaceGravity2D.SimulationControl.minAttractionRange  [get], [set]
```

Global constraint for gravitational attraction range.

6.9.6.14 minAttractorMass

```
double SpaceGravity2D.SimulationControl.minAttractorMass  [get], [set]
```

Mass threshold for body to became attractor

6.9.6.15 timeScale

```
double SpaceGravity2D.SimulationControl.timeScale  [get], [set]
```

TimeScale of simulation process. May be dynamicaly changed, but very large values decreasing precision of calculations

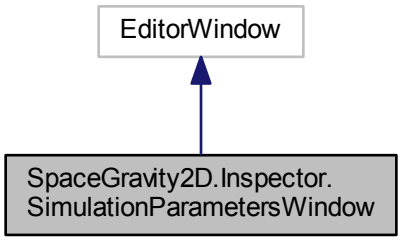
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/SimulationControl.cs

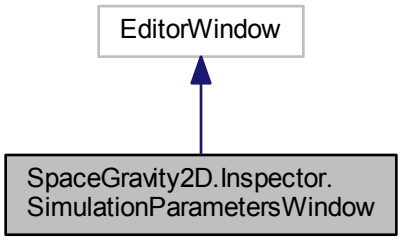
6.10 SpaceGravity2D.Inspector.SimulationParametersWindow Class Reference

Unity editor window for SimulationControl settings for current scene. Also manages displaying orbits path lines and other tools in sceneview window.

Inheritance diagram for SpaceGravity2D.Inspector.SimulationParametersWindow:



Collaboration diagram for SpaceGravity2D.Inspector.SimulationParametersWindow:



Static Public Member Functions

- static void ShowWindow ()

  Create new window.
- static SimulationControl FindSimulationControlGameObject ()

  Find SimulationControl reference on scene.
- static SimulationControl CreateSimulationControl ()

  Create new simulation control gameobject.
- static void InverseVelocityFor (GameObject[ ] objects)

  Tool: inverse velocity of selection.

6.10.1 Detailed Description

Unity editor window for SimulationControl settings for current scene. Also manages displaying orbits path lines and other tools in sceneview window.

6.10.2 Member Function Documentation

6.10.2.1 CreateSimulationControl()

```
static SimulationControl SpaceGravity2D.Inspector.SimulationParametersWindow.CreateSimulation↩
Control ( )  [static]
```

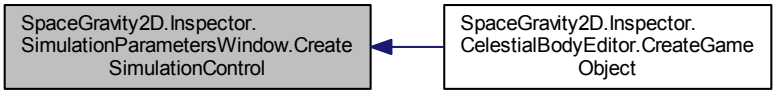
Create new simulation control gameobject.

Returns

Created object reference.

Here is the call graph for this function:

Here is the caller graph for this function:



6.10.2.2 FindSimulationControlGameObject()

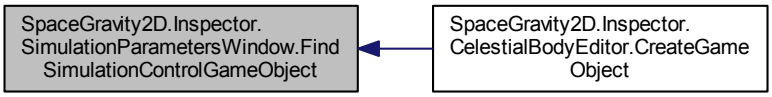
```
static SimulationControl SpaceGravity2D.Inspector.SimulationParametersWindow.FindSimulation↩
ControlGameObject ( )  [static]
```

Find SimulationControl reference on scene.

Returns

Object reference or null.

Here is the caller graph for this function:



6.10.2.3 InverseVelocityFor()

```
static void SpaceGravity2D.Inspector.SimulationParametersWindow.InverseVelocityFor (
            GameObject [] objects )  [static]
```

Tool: inverse velocity of selection.

6.10.2.4 ShowWindow()

```
static void SpaceGravity2D.Inspector.SimulationParametersWindow.ShowWindow ( )  [static]
```

Create new window.

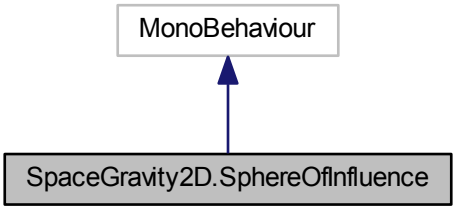
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Editor/SimulationParametersWindow.cs

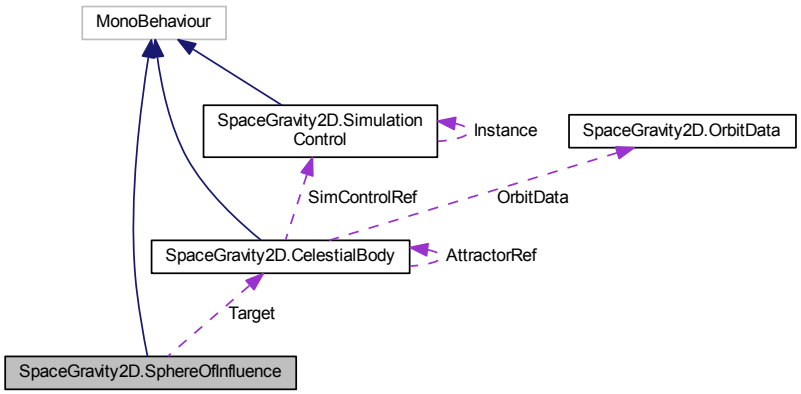
6.11 SpaceGravity2D.SphereOfInfluence Class Reference

Basic static Sphere of Influence component script, alternative to Dynamic Attractor Changing. If attached to gameobject whith no colliders, new collider will be created.

Inheritance diagram for SpaceGravity2D.SphereOfInfluence:



Collaboration diagram for SpaceGravity2D.SphereOfInfluence:



Public Attributes

- SphereCollider Detector

  Reference to collider.
- CelestialBody Target

  Reference to celestial body.
- float TriggerRadius

  Radius of collider detector in world units.
- bool UseAutoROI = false

  If true and attractor is not null, range of influence will be calculated automaticaly. Useful for making first approach.
- bool IgnoreBodiesWithDynamicAttrChanging = true

Dynamic attractor changing of celestial body is full alternative to this component.

- bool IgnoreTransformsScale = true

  Dont affect transform scale on trigger radius.

- bool IgnoreOtherSpheresOfInfluences = true

  Don't trigger by other colliders.

- bool DrawGizmo

  Draw sphere in editor.

6.11.1 Detailed Description

Basic static Sphere of Influence component script, alternative to Dynamic Attractor Changing. If attached to gameobject whith no colliders, new collider will be created.

6.11.2 Member Data Documentation

6.11.2.1 Detector

```
SphereCollider SpaceGravity2D.SphereOfInfluence.Detector
```

Reference to collider.

6.11.2.2 DrawGizmo

```
bool SpaceGravity2D.SphereOfInfluence.DrawGizmo
```

Draw sphere in editor.

6.11.2.3 IgnoreBodiesWithDynamicAttrChanging

```
bool SpaceGravity2D.SphereOfInfluence.IgnoreBodiesWithDynamicAttrChanging = true
```

Dynamic attractor changing of celestial body is full alternative to this component.

6.11.2.4 IgnoreOtherSpheresOfInfluences

```
bool SpaceGravity2D.SphereOfInfluence.IgnoreOtherSpheresOfInfluences = true
```

Don't trigger by other colliders.

6.11.2.5 IgnoreTransformsScale

```
bool SpaceGravity2D.SphereOfInfluence.IgnoreTransformsScale = true
```

Dont affect transform scale on trigger radius.

6.11.2.6 Target

```
CelestialBody SpaceGravity2D.SphereOfInfluence.Target
```

Reference to celestial body.

6.11.2.7 TriggerRadius

```
float SpaceGravity2D.SphereOfInfluence.TriggerRadius
```

Radius of collider detector in world units.

6.11.2.8 UseAutoROI

```
bool SpaceGravity2D.SphereOfInfluence.UseAutoROI = false
```

If true and attractor is not null, range of influence will be calculated automaticaly. Useful for making first approach.

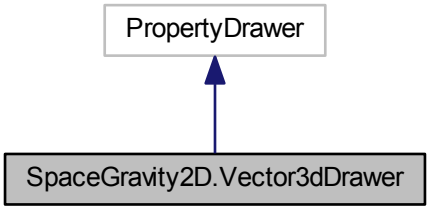
The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Scripts/SphereOfInfluence.cs

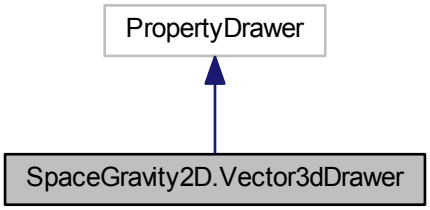
6.12 SpaceGravity2D.Vector3dDrawer Class Reference

Unity editor extension - property drawer for Vector3d type.

Inheritance diagram for SpaceGravity2D.Vector3dDrawer:



Collaboration diagram for SpaceGravity2D.Vector3dDrawer:



Public Member Functions

- override float GetPropertyHeight (SerializedProperty property, GUIContent label)
- override void OnGUI (Rect position, SerializedProperty property, GUIContent label)

6.12.1 Detailed Description

Unity editor extension - property drawer for Vector3d type.

See also

UnityEditor.PropertyDrawer

6.12.2 Member Function Documentation

6.12.2.1 GetPropertyHeight()

```
override float SpaceGravity2D.Vector3dDrawer.GetPropertyHeight (
            SerializedProperty property,
            GUIContent label )
```

6.12.2.2 OnGUI()

```
override void SpaceGravity2D.Vector3dDrawer.OnGUI (
            Rect position,
            SerializedProperty property,
            GUIContent label )
```

The documentation for this class was generated from the following file:

- C:/SpaceGravity2D/Assets/SpaceGravity2D/Editor/Vector3dDrawer.cs

Index