DPhysics Documentation

John Pan
Version 1.0
5/5/2015

Class Index

Class List

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

.

Class Documentation

DPhysics.Body Class Reference

Body reperesents an object in **DPhysics**.
Inheritance diagram for DPhysics.Body:

Public Member Functions

- void **Offset** (ref **Vector2d** change)
- bool **SetPosition** ()
- bool **SetRotation** ()
- void **Attach** (**Body** child)
- void **Detach** (**Body** child)
- void **SetLocalPosition** (ref **Vector2d** localposition)
- delegate void **CollisionEvent** (**Body** other)
- delegate void **CollisionStartEvent** (**Body** other)
- delegate void **CollisionEndEvent** (**Body** other)
- void **Initialize** (**Vector2d** position)
  The most important action for a physics object: Initializing it!
- void **StartCollision** (**Body** body)
- void **DoCollision** (**Body** body)
- void **EndCollision** (**Body** body)
- void **ApplyForce** (ref **Vector2d** force)
- void **ApplyVelocity** (ref **Vector2d** vel)
- void **ApplyRotationalVelocity** (**FInt** vel)
- void **Visualize** (float LerpTime)

Public Attributes

- bool **IsTrigger** = false
- int **Mass** = 1
  The mass of the object. Note that a mass of 0 will make the object kinematic and unaffected by physics forces.
- **DCollider dCollider**
- **Vector2d Velocity**
- **Vector2d LastVelocity**
- **FInt Speed**
- **FInt RotationalVelocity**
- bool **Changed** = true
- int **ActedCount** = DPhysicsManager.MaxActionsUpon
- bool **PositionChanged**
- bool **PositionChangedBuffer**
- Dictionary< ushort, **CollisionPair** > **MyPairs**
  *This represents all collision calculations this **Body** is responsible for. It is stored within the **Body** for performance reasons.*
- bool **RotationChanged** = true
- HashSet< **Body** > **Children**
- **Body Parent**

- CollisionEndEvent **OnCollideEnd**
- ushort **SimID**

Protected Attributes

- **Vector2d rotation**
- **Vector2d LocalStartRotation**
- **Vector2d LocalStartPosition**

Properties

- bool **Active** `[get, set]`
  *Gets or sets a value indicating whether this **DPhysics.Body** is active.*
- **Vector2d Position** `[get, set]`
- **Vector2d Rotation** `[get, set]`

Events

- CollisionEvent **OnCollide**
- CollisionStartEvent **OnCollideStart**

Detailed Description

Body reperesents an object in **DPhysics**.

Member Function Documentation

void DPhysics.Body.Initialize (Vector2d *position*)

The most important action for a physics object: Initializing it!

Parameters:

position	Position.

Member Data Documentation

int DPhysics.Body.Mass = 1

The mass of the object. Note that a mass of 0 will make the object kinematic and unaffected by physics forces.

Dictionary<ushort,CollisionPair> DPhysics.Body.MyPairs

This represents all collision calculations this **Body** is responsible for. It is stored within the **Body** for performance reasons.

Property Documentation

bool DPhysics.Body.Active `[get]`, `[set]`

Gets or sets a value indicating whether this **DPhysics.Body** is active.

`true` if active; otherwise, `false` .

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

- Body.cs

DPhysics.Bounder Class Reference

Bounder is the 2nd layer of the broad-phase collision detection.

Public Member Functions

- **Bounder** (**DCollider** pol)
- void **BuildBounds** (bool Thorough)
  Builds the bounder of the object.
- void **Offset** (ref **Vector2d** change)

Static Public Member Functions

- static bool **CanIntersect** (**DCollider** polyA, **DCollider** polyB)
  Bounding intersection test if there is a rectangle bound.
- static bool **CanIntersect** (**DCollider** polyA, **DCollider** polyB, ref **FInt** CombinedSqrRadius, out **FInt** sqrdistance)
  Bounding intersection test for only circle bounds.

Public Attributes

- **FInt Radius**
- long **xMax**
- long **xMin**
- long **yMax**
- long **yMin**
- bool **IsCircle** = false

Detailed Description

Bounder is the 2nd layer of the broad-phase collision detection.

Member Function Documentation

void DPhysics.Bounder.BuildBounds (bool *Thorough*)

Builds the bounder of the object.

Parameters:

Thorough	If set to `true` , build completely.

static bool DPhysics.Bounder.CanIntersect (DCollider *polyA*, DCollider *polyB*) `[static]`

Bounding intersection test if there is a rectangle bound.

Returns:

`true` if can intersect the specified polyA polyB; otherwise, `false` .

Parameters:

polyA	Poly a.
polyB	Poly b.

static bool DPhysics.Bounder.CanIntersect (DCollider *polyA*, DCollider *polyB*, ref FInt *CombinedSqrRadius*, out FInt *sqrdistance*) `[static]`

Bounding intersection test for only circle bounds.

Returns:

`true` if can intersect the specified polyA polyB CombinedSqrRadius sqrdistance; otherwise, `false` .

Parameters:

polyA	Poly a.
polyB	Poly b.
CombinedSqrRadius	Combined sqr radius.
sqrdistance	Sqrdistance.

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

- Bounder.cs

DPhysics.CollisionPair Class Reference

Public Member Functions

- **CollisionPair** (**Body** bodyA, **Body** bodyB)
- void **GenerateCollision** ()

Public Attributes

- **Body BodyA**
- **Body BodyB**
- bool **SamePartition**
- bool **Changed**
- readonly bool **SimulatePhysics**
- bool **IsColliding**
- readonly bool **ViableCollision**
- **FInt CombinedSqrRadius**
- **CollisionResult MyCollisionResult**

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

- CollisionPair.cs

DPhysics.CollisionResult Class Reference

Public Member Functions

- **CollisionResult** (**CollisionPair** _pair)
- void **Calculate** ()

Static Public Member Functions

- static **FInt IntervalDistance** (**FInt** minA, **FInt** maxA, **FInt** minB, **FInt** maxB)
- static void **ProjectPolygon** (**Vector2d** axis, **DCollider** dCollider, out **FInt** min, out **FInt** max)

Public Attributes

- bool **Intersect**
- **Vector2d PenetrationVector**
- **Vector2d PenetrationDirection**
- **CollisionPair pair**

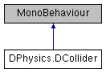
The documentation for this class was generated from the following file:

- CollisionResult.cs

DPhysics.DCollider Class Reference

DCollider represents an object's collider.

Inheritance diagram for DPhysics.DCollider:



Public Member Functions

- void **Initialize** (**Body** body)
  *Initializes this **DCollider**. This is usually called from this physics object's **Body**.*
- void **BuildBounds** ()
  Builds the broad-phase collision bounds of this collider.
- void **BuildEdges** ()
  Builds this object's edges for collision detection.
- void **BuildPoints** ()
- void **Offset** (ref **Vector2d** change)
- override string **ToString** ()

Public Attributes

- **Vector2d center**
  Use of this variable is not supported. Use 'Center' instead.
- Vector2[] **Vertices** = new Vector2[0]
  The original vertices set in the inspector.
- **Vector2d**[] **Points**
  *Vertices of this **DCollider** rotated but not offset.*
- **Vector2d**[] **points**
  *Vertices of this **DCollider** rotated and offset.*
- **Vector2d**[] **backupPoints**
  *Vertices of this **DCollider**, neither rotated nor offset.*
- bool **IsCircle** = false
  Defines whether or not this object is a circle.
- double **Radius**
  The radius of this object set in the inspector.
- **FInt radius**
  The radius of this object used in the simulation.
- **Bounder MyBounds**

Properties

- **Vector2d Center** `[get, set]`
  Gets or sets the center.
- **Vector2d Rotation** `[get, set]`
  Gets or sets the rotation.
- **Vector2d**[] **Edges** `[get]`

Detailed Description

DCollider represents an object's collider.

Member Function Documentation

void DPhysics.DCollider.BuildBounds ()

Builds the broad-phase collision bounds of this collider.

void DPhysics.DCollider.BuildEdges ()

Builds this object's edges for collision detection.

void DPhysics.DCollider.Initialize (Body *body*)

Initializes this **DCollider**. This is usually called from this physics object's **Body**.

Parameters:

body	**This DCollider's Body**

Member Data Documentation

Vector2d [] DPhysics.DCollider.backupPoints

Vertices of this **DCollider**, neither rotated nor offset.

Vector2d DPhysics.DCollider.center

Use of this variable is not supported. Use 'Center' instead.

bool DPhysics.DCollider.IsCircle = false

Defines whether or not this object is a circle.

Vector2d [] DPhysics.DCollider.Points

Vertices of this **DCollider** rotated but not offset.

Vector2d [] DPhysics.DCollider.points

Vertices of this **DCollider** rotated and offset.

double DPhysics.DCollider.Radius

The radius of this object set in the inspector.

FInt DPhysics.DCollider.radius

The radius of this object used in the simulation.

Vector2 [] DPhysics.DCollider.Vertices = new Vector2[0]

The original vertices set in the inspector.

Property Documentation

Vector2d DPhysics.DCollider.Center `[get], [set]`

Gets or sets the center.

The center.

Vector2d DPhysics.DCollider.Rotation `[get], [set]`

Gets or sets the rotation.

The rotation.

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

- DCollider.cs

FInt Struct Reference

Public Member Functions

- void **Multiply** (long OtherRawValue, out **FInt** ret)
- void **Multiply** (int OtherValue, out **FInt** ret)
- void **Divide** (long OtherRawValue, out **FInt** ret)
- void **Divide** (int OtherValue, out **FInt** ret)
- void **Modulo** (long OtherRawValue, out **FInt** ret)
- void **Modulo** (int OtherValue, out **FInt** ret)
- void **Add** (long OtherRawValue, out **FInt** ret)
- void **Add** (int OtherValue, out **FInt** ret)
- void **Subtract** (long OtherRawValue, out **FInt** ret)
- void **Subtract** (int OtherValue, out **FInt** ret)
- bool **Equals** (long OtherRawValue)
- bool **MoreEquals** (long OtherRawValue)
- bool **LessEquals** (long OtherRawValue)
- bool **More** (long OtherRawValue)
- bool **Less** (long OtherRawValue)
- bool **AbsoluteValueMoreThan** (long OtherRawValue)
- void **AbsoluteValue** (out **FInt** ret)
- void **Sign** ()
- void **Inverse** (out **FInt** ret)
- float **ToFloat** ()
- int **ToInt** ()
- double **ToDouble** ()
- short **ToRoundedShort** ()
- override int **GetHashCode** ()
- override string **ToString** ()

Static Public Member Functions

- static **FInt Create** (int StartingValue)
  *Creates an **FInt** with an equaivalent value of StartingValue*
- static **FInt Create** (long StartingRawValue)
  *Creates an **FInt** with a raw value of StartingRawValue. A StartingRawValue of 1 is equal to 2 ^ -SHIFT_AMOUNT.*
- static **FInt Create** (float FloatValue)
  *Create an **FInt** from a float.*
- static **FInt Create** (double DoubleValue)
  *Creates an **FInt** from a double.*
- static **FInt FromParts** (long PreDecimal, long PostDecimal)
- static **FInt operator*** (**FInt** one, **FInt** other)
- static **FInt operator*** (**FInt** one, int multi)
- static **FInt operator/** (**FInt** one, **FInt** other)
- static **FInt operator/** (**FInt** one, int divisor)
- static **FInt operator%** (**FInt** one, **FInt** other)
- static **FInt operator+** (**FInt** one, **FInt** other)
- static **FInt operator+** (**FInt** one, int other)
- static **FInt operator-** (**FInt** one, **FInt** other)
- static **FInt operator-** (**FInt** one, int other)

- static bool **operator==** (**FInt** one, **FInt** other)
- static bool **operator!=** (**FInt** one, **FInt** other)
- static **FInt operator-** (**FInt** src)

Public Attributes

- long **RawValue**
- const int **SHIFT_AMOUNT** = 24
  This represents the resolution of integers. A higher value makes calculations more accurate but reduces performance slightly. A higher value also has the potential to cause overflow under extreme conditions. Note: Must be even.
- const long **MAX_VALUE** = long.MaxValue >> **SHIFT_AMOUNT**
  Represents the maximum value, usable before run-time.
- const long **OneRaw** = (long)1 << **SHIFT_AMOUNT**

Static Public Attributes

- static **FInt OneF** = **FInt.Create**(1)
- static **FInt TwoF** = **FInt.Create** (2)
- static **FInt ZeroF** = **FInt.Create** (0)
- static **FInt HalfF** = **FInt.Create**((long)(1 << (**SHIFT_AMOUNT** - 1)))
- static **FInt MaxValue** = **FInt.Create**(long.MaxValue)

Member Function Documentation

static FInt FInt.Create (int *StartingValue*) `[static]`

Creates an **FInt** with an equaivalent value of StartingValue

Parameters:

StartingValue	Starting value.

static FInt FInt.Create (long *StartingRawValue*) `[static]`

Creates an **FInt** with a raw value of StartingRawValue. A StartingRawValue of 1 is equal to 2 ^ -SHIFT_AMOUNT.

Parameters:

StartingRawValue	Starting raw value.

static FInt FInt.Create (float *FloatValue*) `[static]`

Create an **FInt** from a float.

Parameters:

FloatValue	Float value.

static FInt FInt.Create (double *DoubleValue*) [static]

Creates an **FInt** from a double.

Parameters:

DoubleValue	Double value.

Member Data Documentation

const long FInt.MAX_VALUE = long.MaxValue >> SHIFT_AMOUNT

Represents the maximum value, usable before run-time.

const int FInt.SHIFT_AMOUNT = 24

This represents the resolution of integers. A higher value makes calculations more accurate but reduces performance slightly. A higher value also has the potential to cause overflow under extreme conditions. Note: Must be even.

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

- Math/FInt.cs

Vector2d Struct Reference

Vector2d is the fixed-math equaivalent of Vector2.

Public Member Functions

- **FInt Magnitude** (out **FInt** ret)
  *The magnitude of this **Vector2d**.*
- **FInt SqrMagnitude** (out **FInt** ret)
  *The Square Magnitude of this **Vector2d**.*
- **Vector2d** (**FInt** x, **FInt** y)
  *Initializes a new instance of **Vector2d** with fixed-point values.*
- **Vector2d** (long xInt, long yInt)
  *Initializes a new instance of **Vector2d** with integer values.*
- void **Add** (ref **Vector2d** Other, out **Vector2d** ret)
- void **Subtract** (ref **Vector2d** Other, out **Vector2d** ret)
- void **Multiply** (long OtherRawValue, out **Vector2d** ret)
- void **Multiply** (int OtherValue, out **Vector2d** ret)
- void **Divide** (long OtherRawValue, out **Vector2d** ret)
- void **Divide** (int OtherValue, out **Vector2d** ret)
- void **Invert** ()
- void **Normalize** ()
  Normalizes this vector to have a magnitude of 1. Note: This is very optimized and yields better results than dividing by the Magnitdue.
- Vector2 **ToSinglePrecision** ()
  *Converts this **Vector2d** into a Vector2*
- override string **ToString** ()
- override int **GetHashCode** ()
- bool **Equals** (ref **Vector2d** other)
- void **Rotate** (long CosRaw, long SinRaw, out **Vector2d** ret)
  *Rotates the object by an angle whose Cosine and Sine raw values eequal to CosRaw and SinRaw. Note: **DPhysics** rotations start at the top of the unit circle. If you're using standard rotations, get the localright of the rotated Vector.*
- void **RotateTowards** (ref **Vector2d** target, **FInt** amount, out **Vector2d** ret)
  Increments this rotation towards a target rotation. Note: target must be a unit vector.

Static Public Member Functions

- static **operator Vector2d** (Vector3 v)
- static **operator Vector3** (**Vector2d** v)
- static **Vector2d operator+** (**Vector2d** a, **Vector2d** b)
- static **Vector2d operator-** (**Vector2d** a, **Vector2d** b)
- static **Vector2d operator-** (**Vector2d** a)
- static **Vector2d operator*** (**Vector2d** a, **FInt** d)
- static **Vector2d operator*** (**Vector2d** a, int d)
- static **Vector2d operator/** (**Vector2d** a, **FInt** d)
- static **Vector2d operator/** (**Vector2d** a, int d)
- static void **Dot** (ref **Vector2d** lhs, ref **Vector2d** rhs, out **FInt** ret)
  Returns the dot product of 2 Vector2ds.
- static void **Cross** (ref **Vector2d** U, ref **Vector2d** B, out **FInt** ret)

Retursn the cross product of 2 Vector2ds.

- static void **Reflect** (ref **Vector2d** vector, ref **Vector2d** normal, out **Vector2d** ret)
  Reflects vector over the axis of Normal. Note: Normal must be normalized.

Public Attributes

- **FInt x**
- **FInt y**

Static Public Attributes

- static **Vector2d zero** = new **Vector2d**(0,0)
  Value of (0,0)
- static **Vector2d one** = new **Vector2d**(1,1)
  Value of (1,1)
- static **Vector2d up** = new **Vector2d**(0,1)
  Value of (0,1)
- static **Vector2d right** = new **Vector2d**(1,0)
  Value of (1,0)

Properties

- **FInt this[int index]** `[get, set]`
- **Vector2d localright** `[get]`
  Returns the vector orthogonal and to the right of this vector.

Detailed Description

Vector2d is the fixed-math equaivalent of Vector2.

Constructor & Destructor Documentation

Vector2d.Vector2d (FInt *x*, FInt *y*)

Initializes a new instance of **Vector2d** with fixed-point values.

Parameters:

x	The x coordinate.
y	The y coordinate.

Vector2d.Vector2d (long *xInt*, long *yInt*)

Initializes a new instance of **Vector2d** with integer values.

Parameters:

xInt	X int.
yInt	Y int.

Member Function Documentation

static void Vector2d.Cross (ref Vector2d *U*, ref Vector2d *B*, out FInt *ret*) `[static]`

Retursn the cross product of 2 Vector2ds.

Parameters:

U	U.
B	B.
ret	Ret.

static void Vector2d.Dot (ref Vector2d *lhs*, ref Vector2d *rhs*, out FInt *ret*) `[static]`

Returns the dot product of 2 Vector2ds.

Parameters:

lhs	Lhs.
rhs	Rhs.
ret	Ret.

FInt Vector2d.Magnitude (out FInt *ret*)

The magnitude of this **Vector2d**.

Parameters:

ret	Ret.

void Vector2d.Normalize ()

Normalizes this vector to have a magnitude of 1. Note: This is very optimized and yields better results than dividing by the Magnitdue.

static void Vector2d.Reflect (ref Vector2d *vector*, ref Vector2d *normal*, out Vector2d *ret*) `[static]`

Reflects vector over the axis of Normal. Note: Normal must be normalized.

Parameters:

vector	Vector.
normal	Normal.
ret	Ret.

void Vector2d.Rotate (long *CosRaw*, long *SinRaw*, out Vector2d *ret*)

Rotates the object by an angle whose Cosine and Sine raw values eequal to CosRaw and SinRaw. Note: **DPhysics** rotations start at the top of the unit circle. If you're using standard rotations, get the localright of the rotated Vector.

Parameters:

CosRaw	Cos raw.
SinRaw	Sin raw.
ret	Ret.

void Vector2d.RotateTowards (ref Vector2d *target*, FInt *amount*, out Vector2d *ret*)

Increments this rotation towards a target rotation. Note: target must be a unit vector.

Parameters:

target	Target.
amount	Amount.
ret	Ret.

FInt Vector2d.SqrMagnitude (out FInt *ret*)

The Square Magnitude of this **Vector2d**.

Returns:

The magnitude.

Parameters:

ret	Ret.

Vector2 Vector2d.ToSinglePrecision ()

Converts this **Vector2d** into a Vector2

Returns:

The single precision.

Member Data Documentation

Vector2d Vector2d.one = new Vector2d(1,1) [static]

Value of (1,1)

Vector2d Vector2d.right = new Vector2d(1,0) [static]

Value of (1,0)

Vector2d Vector2d.up = new Vector2d(0,1) [static]

Value of (0,1)

Vector2d Vector2d.zero = new Vector2d(0,0) [static]

Value of (0,0)

Property Documentation

Vector2d Vector2d.localright [get]

Returns the vector orthogonal and to the right of this vector.

The localright.

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

- Math/Vector2d.cs