96PixelGames Documentation

Release 1.0

Sven Hecht

January 17, 2017

Contents

Contents:

CHAPTER 1

96 Pixel Games SDK

The 96 Pixel Games SDK

Author: Sven Hecht <sven.hecht@gmail.com>

1.1 Sub Packages

1.1.1 Sprites

A Package containing prefab sprites

Explosion module

class gamelib.sprites.explosion.**Explosion**(*position*)
Bases: *gamelib.sprite.Sprite*

gamelib.sprites.explosion.**getrandbits**(*k*) → x. Generates a long int with k random bits.

gamelib.sprites.explosion.**random**() → x in the interval [0, 1).

1.2 animatedgameobject

class gamelib.animatedgameobject.**AnimatedGameObject**(*position*, *color1*, *color2=None*, *animationDuration=1*, *loop=1*, *algorithm=<function Linear>*)
Bases: *gamelib.gameobject.GameObject*

The AnimatedGameObject is able to let a pixel animate between two colors.

draw(*rgb*)
The draw method should be called regularily. It draws the next frame

update(*dt*)
The update method should be called regularily. It calculates the next animation step

gamelib.animatedgameobject.**getrandbits**(*k*) → x. Generates a long int with k random bits.

gamelib.animatedgameobject.**random**() → x in the interval [0, 1).

1.3 animation

class gamelib.animation.**Animation**(*startValue*, *endValue*, *duration*, *loop=1*, *algorithm=<function Linear>*)

Bases: object

getValue()

restart()

update(*dt*)

class gamelib.animation.**AnimationAlgorithm**

Bases: object

A List of Animation Algorithms.

Sources: http://wpf-animation.googlecode.com/svn/trunk/src/WPF/Animation/PennerDoubleAnimation.cs https://github.com/danro/jquery-easing/blob/master/jquery.easing.js

static EaseInBounce(*t*, *b*, *c*, *d*)
deceleration until halfway, then acceleration.

static EaseInCubic(*t*, *c*, *b*, *d*)
accelerating from zero velocity

static EaseInElastic(*t*, *c*, *b*, *d*)

static EaseInOutCubic(*t*, *c*, *b*, *d*)
acceleration until halfway, then deceleration

static EaseInOutQuad(*t*, *c*, *b*, *d*)
acceleration until halfway, then deceleration

static EaseInOutQuart(*t*, *c*, *b*, *d*)
acceleration until halfway, then deceleration

static EaseInOutQuint(*t*, *c*, *b*, *d*)
acceleration until halfway, then deceleration

static EaseInQuad(*t*, *c*, *b*, *d*)
accelerating from zero velocity

static EaseInQuart(*t*, *c*, *b*, *d*)
accelerating from zero velocity

static EaseInQuint(*t*, *c*, *b*, *d*)
accelerating from zero velocity

static EaseOutBounce(*t*, *c*, *b*, *d*)
exponentially decaying parabolic bounce

static EaseOutCubic(*t*, *c*, *b*, *d*)
decelerating to zero velocity !!! BROKEN !!!

static EaseOutElastic(*t*, *c*, *b*, *d*)

static EaseOutQuad(*t*, *c*, *b*, *d*)
decelerating to zero velocity

static EaseOutQuart(*t*, *c*, *b*, *d*)
decelerating to zero velocity

static EaseOutQuint(*t*, *c*, *b*, *d*)
decelerating to zero velocity !!! BROKEN !!!

static Linear(*t*, *c*, *b*, *d*)
no easing, no acceleration

class gamelib.animation.**AnimationDirection**
Bases: object

Backward = 1

Forward = 0

class gamelib.animation.**AnimationLoopType**
Bases: object

Loop = 1

OneTime = 0

PingPong = 2

1.4 color

class gamelib.color.**Color**
Bases: object

static add(*color*, *other_color*)

static multiply(*color*, *factor*)

1.5 game

1.6 gameobject

gameobject.py: Represents an object in a game.

class gamelib.gameobject.**GameObject**
Bases: object

The GameObject class represents an object in a game. It is mostly used as a base class for other elements that enrich it's capabilities.

If used should be updated and drawn.

draw(*rgb*)

update(*dt*)

gamelib.gameobject.**getrandbits**(*k*) → x. Generates a long int with k random bits.

gamelib.gameobject.**random**() → x in the interval [0, 1).

1.7 keyboardcontroller

1.8 librgb

class gamelib.librgb.**RGB**(*ip=None*, *port=6803*, *verbose=False*)
Bases: object

add_color(*v*, *color*)

clear(*color*)

mix_color(*v*, *color*, *alpha*)

send()

setPixel(*v*, *color*)

class gamelib.librgb.**SPIDevice**(*device='/dev/spidev0.0'*)
Bases: object

init()

tick()

write(*buffer*)

gamelib.librgb.**clampByte**(*i*)

gamelib.librgb.**getrandbits**(*k*) → x. Generates a long int with k random bits.

gamelib.librgb.**random**() → x in the interval [0, 1).

1.9 menu

class gamelib.menu.**Menu**(*game*, *name*, *items*)
Bases: *gamelib.state.State*

draw(*rgb*)

layoutMenuItem(*index*, *name*, *itemCount*)

onAxisChanged(*player*, *xAxis*, *yAxis*, *previousXAxis*, *previousYAxis*)

onButtonChanged(*player*, *aButton*, *bButton*, *previousAButton*, *previousBButton*)

onClampedAxisChanged(*player*, *x*, *y*)

onEnter(*oldState*)

onItemClicked(*name*)

onLeave(*newState*)

update(*dt*)

class gamelib.menu.**MenuItem**(*index*, *name*, *position*, *size*)
Bases: *gamelib.animatedgameobject.AnimatedGameObject*

draw(*rgb*)

gamelib.menu.**getrandbits**(*k*) → x. Generates a long int with k random bits.

gamelib.menu.**random**() → x in the interval [0, 1).

1.10 music

1.11 resource

class gamelib.resource.**Resource**(*name*, *resFile*)
Bases: object

load()

1.12 sound

1.13 sprite

class gamelib.sprite.**Sprite**(*spriteData*, *duration*, *loop*)
Bases: *gamelib.gameobject.GameObject*

draw(*rgb*)

update(*dt*)

gamelib.sprite.**getrandbits**(*k*) → x. Generates a long int with k random bits.

gamelib.sprite.**random**() → x in the interval [0, 1).

1.14 state

class gamelib.state.**State**(*name*)
Bases: object

draw(*rgb*)

onAxisChanged(*player*, *xAxis*, *yAxis*, *previousXAxis*, *previousYAxis*)

onButtonChanged(*player*, *aButton*, *bButton*, *previousAButton*, *previousBButton*)

onClampedAxisChanged(*player*, *x*, *y*)

onEnter(*oldState*)

onLeave(*newState*)

set_game(*game*)

update(*dt*)

1.15 statemachine

class gamelib.statemachine.**StateChange**
Bases: object

Enter = 1

Leave = 2

Unknown = 0

class gamelib.statemachine.**StateMachine**(*game*, *states*, *state_change_callback=None*)
Bases: object

draw(*rgb*)

onAxisChanged(*player*, *x_axis*, *y_axis*, *previous_x_axis*, *previous_y_axis*)

onButtonChanged(*player*, *a_button*, *b_button*, *previous_a_button*, *previous_b_button*)

onClampedAxisChanged(*player*, *x*, *y*)

setState(*name*)

update(*dt*)

1.16 vector

vector.py: A simple little Vector class. Enabling basic 2D vector math.

class gamelib.vector.**Vector**(*x=0*, *y=0*)
Bases: object

The Vector class can represent a direction or a position in 2-dimensional space

Examples

(Vector(2, 5) + Vector(3, 1.5)).getNormalized()

Parameters

- **x** (*number/tuple/list/Vector, Optional*) – Represents the x dimension of the vector. If the first argument is a Vector, tuple or list the x and y dimensions will be initialized. The default x value is 0
- **y** (*number, Optional*) – Represents the y dimension of the vector. The default y value is 0

__add__(*other*)
Calculates the sum between this vector and the given value.

Examples

Vector(16, 22) + 5 => Vector(21, 27) Vector(13, 13) + Vector(7, 5) => Vector(20, 18) Vector(13, 13) + [7, 5] => Vector(20, 18) Vector(13, 13) + (7, 5) => Vector(20, 18)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the add function with

Returns A new Vector instance containing the sum of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__div__(*other*)
Calculates the quotient between this vector and the given value.

Examples

Vector(16, 22) / 5 => Vector(3, 4) Vector(12, 13) / Vector(6, 5) => Vector(2, 2) Vector(12, 13) / [6, 5] => Vector(2, 2) Vector(12, 13) / (6, 5) => Vector(2, 2) Vector(12.0, 13.0) / (6, 5) => Vector(2.0, 2.6)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the division function with

Returns A new Vector instance containing the quotient of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__eq__(*other*)
The Equality comparer.

Examples

Vector(6, 2) == Vector(6, 2) => True Vector(6, 2) == Vector(6, 3) => False

Parameters **other** (Vector) – the other Vector to compare this one to

Returns a bool representing the result of the comparison

Return type Boolean

Raises NotImplemented for arguments of not accepted type

__ge__(*other*)
The greater than equals comparer.

Examples

Vector(6, 2) >= Vector(5, 1) => True Vector(6, 2) >= Vector(7, 3) => False Vector(6, 2) >= Vector(6, 2) => True

Parameters **other** (Vector) – the other Vector to compare this one to

Returns a bool representing the result of the comparison

Return type Boolean

Raises NotImplemented for arguments of not accepted type

__gt__(*other*)
The greater than comparer.

Examples

Vector(6, 2) > Vector(5, 1) => True Vector(6, 2) > Vector(7, 3) => False Vector(6, 2) > Vector(6, 2) => False

Parameters **other** (Vector) – the other Vector to compare this one to

Returns a bool representing the result of the comparison

Return type Boolean

Raises NotImplemented for arguments of not accepted type

__iadd__(*other*)
Calculates the sum between this vector and the given value in place.

Examples

Vector(16, 22) += 5 => Vector(21, 27) Vector(13, 13) += Vector(7, 5) => Vector(20, 18) Vector(13, 13) += [7, 5] => Vector(20, 18) Vector(13, 13) += (7, 5) => Vector(20, 18)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the add function with

Returns itself, containing the sum of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__idiv__(*other*)
Calculates the quotient between this vector and the given value in place.

Examples

Vector(16, 22) /= 5 => Vector(3, 4) Vector(12, 13) /= Vector(6, 5) => Vector(2, 2) Vector(12, 13) /= [6, 5] => Vector(2, 2) Vector(12, 13) /= (6, 5) => Vector(2, 2) Vector(12.0, 13.0) /= (6, 5) => Vector(2.0, 2.6)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the division function with

Returns itself, containing the quotient of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__imul__(*other*)
Calculates the product between this vector and the given value in place.

Examples

Vector(16, 22) *= 5 => Vector(80, 110) Vector(13, 13) *= Vector(7, 5) => Vector(91, 65) Vector(13, 13) *= [7, 5] => Vector(91, 65) Vector(13, 13) *= (7, 5) => Vector(91, 65)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the multiply function with

Returns itself, containing the product of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__ipow__(*other*)
Calculates the power of this vector by the given value in place.

Examples

Vector(6, 2) **= 5 => Vector(7776, 32) Vector(6, 2) **= 5.0 => Vector(7776.0, 32.0)

Parameters **other** (*number*) – the value to perform the exponentiation with

Returns itself, containing the result of the calculation

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__isub__(*other*)

Calculates the difference between this vector and the given value in place.

Examples: Vector(16, 22) -= 5 => Vector(11, 17) Vector(13, 13) -= Vector(7, 5) => Vector(6, 8) Vector(13, 13) -= [7, 5] => Vector(6, 8) Vector(13, 13) -= (7, 5) => Vector(6, 8)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the substract function with

Returns itself, containing the difference of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__le__(*other*)

The less than equals comparer.

Examples

Vector(6, 2) <= Vector(5, 1) => False Vector(6, 2) <= Vector(7, 3) => True Vector(6, 2) <= Vector(6, 2) => True

Parameters **other** (Vector) – the other Vector to compare this one to

Returns a bool representing the result of the comparison

Return type Boolean

Raises NotImplemented for arguments of not accepted type

__len__()

Calculates the magnitude of the vector.

Examples

len(Vector(1, 0)) => 1 len(Vector(0, 2)) => 2

Parameters **other** (Vector) – the other Vector to compare this one to

Returns The magnitude of the vector

Return type Number

__lt__(*other*)

The less than comparer.

Examples

Vector(6, 2) < Vector(5, 1) => False Vector(6, 2) < Vector(7, 3) => True Vector(6, 2) < Vector(6, 2) => False

Parameters **other** (Vector) – the other Vector to compare this one to

Returns a bool representing the result of the comparison

Return type Boolean

Raises NotImplemented for arguments of not accepted type

__mul__(*other*)
Calculates the product between this vector and the given value.

Examples

Vector(16, 22) * 5 => Vector(80, 110) Vector(13, 13) * Vector(7, 5) => Vector(91, 65) Vector(13, 13) * [7, 5] => Vector(91, 65) Vector(13, 13) * (7, 5) => Vector(91, 65)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the multiply function with

Returns A new Vector instance containing the product of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__ne__(*other*)
The Unequality comparer.

Examples

Vector(6, 2) != Vector(6, 2) => False Vector(6, 2) != Vector(6, 3) => True

Parameters **other** (Vector) – the other Vector to compare this one to

Returns a bool representing the result of the comparison

Return type Boolean

Raises NotImplemented for arguments of not accepted type

__pow__(*other*)
Calculates the power of this vector by the given value.

Examples

Vector(6, 2) ** 5 => Vector(7776, 32) Vector(6, 2) ** 5.0 => Vector(7776.0, 32.0)

Parameters **other** (*number*) – the value to perform the exponentiation with

Returns A new Vector instance containing the result of the calculation

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__rdiv__(*other*)
Calculates the quotient between the given value and this vector.

Examples

22.0 / Vector(16, 22) => Vector(1.375, 1.0) Vector(22.0, 25.0) / Vector(16, 5) => Vector(1.375, 5.0) [22.0, 25.0] / Vector(16, 5) => Vector(1.375, 5.0) (22.0, 25.0) / Vector(16, 5) => Vector(1.375, 5.0) (22, 25) / Vector(16, 5) => Vector(1, 5)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the division function with

Returns A new Vector instance containing the quotient of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__rsub__(*other*)
Calculates the difference between the given value and this vector.

Examples

5 - Vector(16, 22) => Vector(-11, -17) Vector(7, 5) - Vector(13, 13) => Vector(-6, -8) [7, 5] - Vector(13, 13) => Vector(-6, -8) (7, 5) - Vector(13, 13) => Vector(-6, -8)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the substract function with

Returns A new Vector instance containing the difference of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

__sub__(*other*)
Calculates the difference between this vector and the given value.

Examples

Vector(16, 22) - 5 => Vector(11, 17) Vector(13, 13) - Vector(7, 5) => Vector(6, 8) Vector(13, 13) - [7, 5] => Vector(6, 8) Vector(13, 13) - (7, 5) => Vector(6, 8)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the substract function with

Returns A new Vector instance containing the difference of the values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

static angle(*v1*, *v2*)
Calculates the angle in Radian between 2 Vectors

Parameters

- **a** (Vector) – first vector
- **b** (Vector) – second vector

Returns a number in radian representing the angle between the to vectors.

Return type Number

static angleDeg(*v1*, *v2*)
Calculates the angle in Degree between 2 Vectors

Parameters

- **a** (Vector) – first vector
- **b** (Vector) – second vector

Returns a number in degree representing the angle between the to vectors.

Return type Number

clone()
Clones the current Vector

Returns A new instance of the Vector

Return type *Vector*

static distance(*a*, *b*)
Calculates the distance between 2 Vectors

Parameters

- **a** (Vector) – the "from" point
- **b** (Vector) – the "to" point

Returns a number representing the distance between the to vectors (if they represent points in space)

Return type Number

dotproduct(*other*)
Calculates the dot product between this vector and the given value.

Parameters **other** (*Vector/tuple/list*) – the value to perform the dot product function with

Returns A new Vector instance containing the calculated values

Return type *Vector*

Raises NotImplemented for arguments of not accepted type

getLength()
Calculates the magnitude of the vector.

Examples

Vector(1, 0).getLength() => 1 Vector(0, 2).getLength() => 2

Parameters **other** (Vector) – the other Vector to compare this one to

Returns The magnitude of the vector

Return type Number

getNormalized()
Creates a new normalized instance of the Vector

Returns A new instance of type Vector but normalized

Return type *Vector*

modulo(*other*)
Calculates the modulo between this vector and the given value.

Examples

Vector(16, 22).modulo(5) => Vector(1, 2) Vector(13, 13).modulo(Vector(7, 5)) => Vector(6, 3) Vector(13, 13).modulo([7, 5]) => Vector(6, 3) Vector(13, 13).modulo((7, 5)) => Vector(6, 3)

Parameters **other** (*Vector/tuple/list/number*) – the value to perform the modulo function with

Returns A new Vector instance containing the calculated values

Return type *Vector*

static random(*size=1*)
Creates a randomized Vector contained inside a square of the dimensions size x size.

Parameters **size** (*number, Optional*) – Determines the max bounds of the new Random Vector. Default is 1.

Returns a new instance of type Vector

Return type *Vector*

static randomUnitCircle()
Creates a randomized unit Vector that lies on the unit circle (circle of radius 1).

Returns a new instance of type Vector

Return type *Vector*

set(*x*, *y*)
Updates the dimensions of the Vector.

Parameters

- **x** (*number*) – the new value for the x dimension
- **y** (*number*) – the new value for the y dimension

toArr()
Creates an array of the form [x, y]

Returns an array representing the Vector

Return type Array

toInt()
Casts the dimensions to int

Returns a new Vector instance containing integer dimensions

Return type *Vector*

toIntArr()
Casts the dimensions to int and creates an array of the form [x, y]

Returns an new array of the Vectors dimensions casted to integer

Return type *Vector*

gamelib.vector.**getrandbits**(*k*) → x. Generates a long int with k random bits.

gamelib.vector.**random**() → x in the interval [0, 1).

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

g

Symbols

A

B

C

D

E

T

U

V

W