# Brython Crafty Documentation

***Release 0.2.3***

**Carlo E.T. Oliveira**

December 06, 2015

Contents


**1 Brython-Crafty Wrapper** **3**
- 1.1 Introduction . . . 3
- 1.2 Modules . . . 3

**2 Core Module Description** **5**
- 2.1 Crafty . . . 5

**3 Base Module Description** **7**
- 3.1 Base . . . 7
- 3.2 ViewPort . . . 7

**4 Entity Module Description** **9**
- 4.1 Entity . . . 9

**5 Graphics Module Description** **11**
- 5.1 Canvas . . . 11
- 5.2 Sprite . . . 11
- 5.3 Draggable . . . 12

**6 Brython Crafty - API** **13**
- 6.1 Base API . . . 13
- 6.2 Core API . . . 18
- 6.3 Entity API . . . 20
- 6.4 Graphics API . . . 24

**7 Indices and tables** **29**

**Python Module Index** **31**

Contents:

CHAPTER 1

# Brython-Crafty Wrapper

## 1.1 Introduction

Documentation for Crafty.js

This documentation is not meant to document every function there is in Crafty.js, quite opposite! It will be a introduction on the different fields that Crafty.js helps you on, like how you will get started moving things around and reponds to event, when you for an example hit something.

It will make you understand Crafty.js better, and make you going, and perhaps give you the courage to look at the API documentation yourself where every Crafty.js function is documentated.

## 1.2 Modules

Brython-Crafty is a wrapper for Crafty.js developed in Brython

All Brython-Crafty functionality is alloted to single page, documented in *Core Module Description*

CHAPTER 2

# Core Module Description

**See also:**

Module `crafty`

**Note:** Aggregates factory, control and interface units in this single module

## 2.1 Crafty

Set of methods added to every single entity.

**See also:**

Class *crafty.core.BCrafty*

**Note:** Main API Unit.

CHAPTER 3

# Base Module Description

**See also:**

Module `crafty`

**Note:** Aggregates factory, control and interface units in this single module

## 3.1 Base

Set of methods added to Crafty class.

**See also:**

Class *crafty.base.Base*

**Note:** Main Base API Unit.

## 3.2 ViewPort

Manages Camera at 2D games.

**See also:**

Class *crafty.base.ViewPort*

**Note:** Aggregates viewport facilities to Crafty class.

CHAPTER 4

# Entity Module Description

**See also:**

Module *crafty.entity*

**Note:** Aggregates entity and component in this single module

## 4.1 Entity

Creates an entity. Any arguments will be applied in the same way .addComponent() is applied as a quick way to add components.

Any component added will augment the functionality of the created entity by assigning the properties and methods from the component to the entity.

**Example**

```
myEntity = Crafty().e("2D, DOM, Color");
```

**Events**

***NewEntity [Data: { id:Number }]*** When the entity is created and all components are added

See Also

**See also:**

Class *crafty.entity.Entity*

**Note:** Composite Element.

CHAPTER 5

# Graphics Module Description

**See also:**

Module *crafty.graphics*

**Note:** Aggregates canvas and sprite in this single module

## 5.1 Canvas

When this component is added to an entity it will be drawn to the global canvas element. The canvas element (and hence all Canvas entities) is always rendered below any DOM entities.

Crafty.canvas.init() will be automatically called if it is not called already to initialize the canvas element.

Create a canvas entity like this

```
myEntity = Crafty().e("2D, Canvas, Color")\
     .color("green")\
     .attr(x= 13, y= 37, w= 42, h= 42);
```

**Events**

***Draw [Data: {type: "canvas", pos, co, ctx}]*** when the entity is ready to be drawn to the stage

***NoCanvas*** if the browser does not support canvas

**See also:**

Class *crafty.graphics.Canvas*

**Note:** DOm Element Unit.

## 5.2 Sprite

Component for using tiles in a sprite map.

**Events**

***Invalidate*** when the sprites change

**See also:**

Class *crafty.graphics.Sprite*

**Note:** Composite Unit.

## 5.3 Draggable

Enable drag and drop of the entity.

**Events**

***Dragging*** **[Data: MouseEvent]** is triggered each frame the entity is being dragged

***StartDrag*** **[Data: MouseEvent]** is triggered when dragging begins

***StopDrag*** **[Data: MouseEvent]** is triggered when dragging ends

**See also:**

Class *crafty.graphics.Draggable*

**Note:** Interface Unit.

CHAPTER 6

# Brython Crafty - API

## 6.1 Base API

### 6.1.1 Base Module

**Author** *Carlo E. T. Oliveira*

**Contact** carlo@nce.ufrj.br

**Date** 2014/09/23

**Status** This is a "work in progress"

**Revision** 0.1.0

**Home** Labase

**Copyright** 2013, GPL.

**class** `crafty.base.`**`Base`**(*crafty*)
Crafty base operations. *Base*

**Parameters**

- **`w`** – The width of crafty window
- **`h`** – The height of crafty window
- **`stage`** – An element to which this window will be attached

**Returns** An instance of Crafty

**`attr`**(***kwarg*)
Set attributes. *`crafty.entity`*

**Param** kwargs: keyword parameters with name and values of arguments to be changed

**Returns** Self, this same entity

**`background`**(*color*)
Change background color. *`crafty.base.Base`*

**Parameters** **`color`** – A string with components ex:'2D, DOM, Color'

**Returns** This instance of Crafty

**`bind`**(*eventName*, *callback*)
Crafty Bind. *`crafty.base.Base`*

Binds to a global event. Method will be executed when Crafty.trigger is used with the event name.

**Parameters**

- **eventName** – Name of the event to bind to
- **callback** – Method to execute upon event triggered

**Returns** callback function which can be used for unbind

**crafty**()
Crafty js core. *crafty.base.Base*

**Returns** A javascript crafty instance

**destroy**()
Destroy the Entity. *crafty.core.BCrafty* Will remove all event listeners and delete all properties as well as removing from the stage

**Returns** The object destroyied

**isDown**(*keyName*)
Determine if a certain key is currently down. *crafty.base.Base*

**Example**

```
entity.requires('Keyboard').bind('KeyDown', haldle_keydown)
```

Determine if a certain key is currently down. :param keyName: Name or Code of the key to check. See Crafty.keys. :returns: If the key is Down.

**keys**
Keycodes. *crafty.base.Base*

exemple keys.RA keys.LA keys.UA keys. DA

**mousePos**
Mouse Position. *crafty.base.Base*

**onebind**(*eventName*, *callback*)
Crafty OneBind. *crafty.core.BCrafty*

Binds to a global event. Method will be executed once when Crafty.trigger is used with the event name.

**Parameters**

- **eventName** – Name of the event to bind to
- **callback** – Method to execute upon event triggered

**Returns** callback function which can be used for unbind

**text**(*texty*)
Crafty Text. *crafty.base.Base*

String of text that will be inserted into the DOM or Canvas element.

This method will update the text inside the entity.

If you need to reference attributes on the entity itself you can pass a function instead of a string. Example

Crafty.e("2D, DOM, Text").attr({ x: 100, y: 100 }).text("Look at me!!");

**Crafty.e("2D, DOM, Text").attr({ x: 100, y: 100 })** .text(function () { return "My position is " + this._x });

Crafty.e("2D, Canvas, Text").attr({ x: 100, y: 100 }).text("Look at me!!");

**Crafty.e("2D, Canvas, Text").attr({ x: 100, y: 100 })** .text(function () { return "My position is " + this._x });

**Parameters** **text** – Name of the event to bind to

**Returns** Load a crafty scene

**textColor**(*color*)
Change the color of the text. You can use HEX, rgb and rgba colors.

**Parameters** **color** – The color in name, hex, rgb or rgba

**Returns** Self, this same entity

**textFont**(*size='10px'*, *weight='normal'*, *face='normal'*, *family='Arial'*)
Use this method to set font property of the text entity.

**Parameters**

- **size** – Size of the font in pixels ex: "20px"
- **weight** – Weight o font ex: "bold"
- **face** – Type of fonte ex: "italic"
- **family** – Font family

**Returns** Self, this same entity

**unbind**(*eventName*, *callback*)
Crafty unbind. *crafty.core.BCrafty*

Binds to a global event. Method will be executed once when Crafty.trigger is used with the event name.

**Parameters**

- **eventName** – Name of the event to unbind to
- **callback** – Method to unbind

**Returns** True or false depending on if a callback was unbound

**unselectable**()
This method sets the text so that it cannot be selected (highlighted) by dragging.

**Returns** Self, this same entity

**x**
The x position on the stage. *crafty.base.Base*

**y**
The y position on the stage. *crafty.base.Base*

**class** crafty.base.**ViewPort**(*ent*)
Viewport is essentially a 2D camera looking at the stage. Can be moved or zoomed, which in turn will react just like a camera moving in that direction. draggable

**bounds**(*minx*, *miny*, *maxx*, *maxy*)
A rectangle which defines the bounds of the viewport.

**Parameters**

- **minx** – min x bound of viewport
- **miny** – min y bound of viewport
- **maxx** – max x bound of viewport

- **maxy** – max y bound of viewport

**Returns** Self, this same entity

**centerOn** (*target*, *time*)
Centers the viewport on the given entity.

**Parameters**

- **target** – An entity with the 2D component
- **time** – The duration in ms of the camera motion

**Returns** Self, this same entity

**clampToEntities**
Decides if the viewport functions should clamp to game entities. When set to true functions such as Crafty.viewport.mouselook() will not allow you to move the viewport over areas of the game that has no entities. For development it can be useful to set this to false.

**Returns** True if clamped

**follow** (*target*, *offsetx=0*, *offsety=0*)
Follows a given entity with the 2D component. If following target will take a portion of the viewport out of bounds of the world, following will stop until the target moves away.

**Parameters**

- **target** – An entity with the 2D component
- **offsetx** – Follow target should be offsetx pixels away from center
- **offsety** – Positive puts target to the right of center

**Returns** Self, this same entity

**init** (*width*, *height*, *stage_elem*)
Initialize the viewport. If the arguments 'width' or 'height' are missing, use Crafty.DOM.window.width and Crafty.DOM.window.height (full screen model).

The argument 'stage_elem' is used to specify a stage element other than the default, and can be either a string or an HTMLElement. If a string is provided, it will look for an element with that id and, if none exists, create a div. If an HTMLElement is provided, that is used directly. Omitting this argument is the same as passing an id of 'cr-stage'.

**Parameters**

- **width** – Width of the viewport
- **height** – Height of the viewport
- **stage_elem** – the element to use as the stage (either its id or the actual element).

**Returns** Self, this same entity

**mouselook** (*boolean=True*)
Toggle mouselook on the current viewport. Simply call this function and the user will be able to drag the viewport around.

If the user starts a drag, "StopCamera" will be triggered, which will cancel any existing camera animations.

**Parameters** **boolean** – Activate or deactivate mouselook

**Returns** Self, this same entity

**pan** (*dx*, *dy*, *time*)
Pans the camera a given number of pixels over the specified time

**Parameters**

- **dx** – The distance along the x axis
- **dy** – The distance along the y axis
- **time** – The duration in ms for the entire camera movement

**Returns** Self, this same entity

**scale**(*amt*)

Adjusts the scale (zoom). When amt is 1, it is set to the normal scale, e.g. an entity with this.w == 20 would appear exactly 20 pixels wide. When amt is 10, that same entity would appear 200 pixels wide (i.e., zoomed in by a factor of 10), and when amt is 0.1, that same entity would be 2 pixels wide (i.e., zoomed out by a factor of (1 / 0.1)).

If you pass an amt of 0, it is treated the same as passing 1, i.e. the scale is reset.

This method sets the absolute scale, while Crafty.viewport.zoom sets the scale relative to the existing value.

**Parameters** **amt** – amount to zoom in on the target by (eg. 2, 4, 0.5)

**Returns** Self, this same entity

**scroll**(*axis*, *val*)

Will move the viewport to the position given on the specified axis

**Parameters**

- **axis** – 'x' or 'y'
- **val** – The new absolute position on the axis

**Returns** Self, this same entity

**x**

Will move the stage and therefore every visible entity along the x axis in the opposite direction.

**Returns** viewport x

**y**

Will move the stage and therefore every visible entity along the x axis in the opposite direction.

**Returns** viewport y

**zoom**(*amt*, *cent_x*, *cent_y*, *time*)

Zooms the camera in on a given point. amt > 1 will bring the camera closer to the subject amt < 1 will bring it farther away. amt = 0 will reset to the default zoom level Zooming is multiplicative. To reset the zoom amount, pass 0.

**Parameters**

- **amt** – amount to zoom in on the target by (eg. 2, 4, 0.5)
- **cent_x** – the center to zoom on
- **cent_y** – the center to zoom on
- **time** – the duration in ms of the entire zoom operation

**Returns** Self, this same entity

## 6.2 Core API

### 6.2.1 Core Module

**Author** *Carlo E. T. Oliveira*

**Contact** carlo@nce.ufrj.br

**Date** 2014/09/17

**Status** This is a "work in progress"

**Revision** 0.1.0

**Home** Labase

**Copyright** 2013, GPL.

**class** `crafty.core.`**`BCrafty`**(*w=600*, *h=480*, *stage='Document Body'*)
Bases: *crafty.base.Base*, *crafty.base.ViewPort*

Crafty game engine main class. *Crafty*

**Parameters**

- **`w`** – The width of crafty window
- **`h`** – The height of crafty window
- **`stage`** – An element to which this window will be attached

**Returns** An instance of Crafty

**`c`**(*name*, **comp*, ***items*)
Creates a component naming the ID and passing an object. *crafty.core.BCrafty*

A couple of methods are treated specially. They are invoked in partiular contexts, and (in those contexts) cannot be overridden by other components. init will be called when the component is added to an entity remove will be called just before a component is removed, or before an entity is destroyed. It is passed a single boolean parameter that is true if the entity is being destroyed.

**Parameters**

- **`name`** – Name of the component
- **`comp`** – Object with the component's properties and methods that will be inherited by entities.
- **`items`** – If component is not provided each keyword argument will be attached as a member of component.

**`canvas`**()
create a drawing canvas. *crafty.core.BCrafty*

**Returns** A javascript crafty instance

**`crafty`**
Crafty js core. *crafty.core.BCrafty*

**Returns** A javascript crafty instance

**`e`**(*comp='2D, DOM, Color'*)
Entity. *crafty.core.BCrafty*

**Parameters** **`comp`** – A string with components ex:'2D, DOM, Color'

**Returns** An Entity instance

**load**(*name*, *init*)
Crafty Load. *crafty.core.BCrafty*

**Returns** Load a crafty scene

**randRange**(*mini*, *maxi*)
Random Range. *crafty.core.BCrafty*

**Returns** a number ranging from mini to maxi

**scene**(*scene*, *init=None*, *uninit=<function <lambda>>*)
Crafty Scene. *crafty.core.BCrafty*

**Returns** A crafty scene

**sprite**(*x*, *y*, *w*, *h*)
Create a Sprite. *crafty.core.BCrafty*

**Parameters**

- **x** – position x of sprite
- **y** – position y of sprite
- **w** – width w of sprite
- **h** – height h of sprite

**Returns** An instance of Sprite

**spritem**(*tile*, *tileh*, *url*, *paddingX=0*, *paddingY=0*, *paddingAroundBorder=False*, ***mapper*)
Collection of sprites. *crafty.core.BCrafty* Generates components based on positions in a sprite image to be applied to entities.

Accepts a tile size, URL and map for the name of the sprite and its position.

The position must be an array containing the position of the sprite where index 0 is the x position, 1 is the y position and optionally 2 is the width and 3 is the height. If the sprite map has padding, pass the values for the x padding or y padding. If they are the same, just add one value.

If the sprite image has no consistent tile size, 1 or no argument need be passed for tile size.

**Parameters**

- **tile** – Tile size of the sprite map, defaults to 1
- **url** – URL of the sprite image
- **map** – Object where the key is what becomes a new component and the value points to a position on the sprite map
- **paddingX** – Horizontal space in between tiles. Defaults to 0.
- **paddingY** – Vertical space in between tiles. Defaults to paddingX.
- **paddingAroundBorder** – If padding should be applied around the border of the sprite sheet. If enabled the first tile starts at (paddingX,paddingY) instead of (0,0). Defaults to false.

**sprites**(*tile*, *url*, ***mapper*)
Collection of sprites. *crafty.core.BCrafty*

**Parameters**

- **tile** – Tile size of the sprite map, defaults to 1

- **url** – URL of the sprite image
- **map** – Object where the key is what becomes a new component and the value points to a position on the sprite map

**class** crafty.core.**document**

**body = 'Document Body'**

## 6.3 Entity API

### 6.3.1 Entity Module

**Author** *Carlo E. T. Oliveira*

**Contact** carlo@nce.ufrj.br

**Date** 2014/09/17

**Status** This is a "work in progress"

**Revision** 0.1.0

**Home** Labase

**Copyright** 2013, GPL.

**class** crafty.entity.**Entity**(*stage*, *cmp*)
Bases: *crafty.graphics.Sprite*, *crafty.graphics.Draggable*, *crafty.base.Base*

Creates an entity. *Entity*

Any arguments will be applied in the same way .addComponent() is applied as a quick way to add components.

Any component added will augment the functionality of the created entity by assigning the properties and methods from the component to the entity.

**Param** stage: Element to which entity will be attached to

**Param** cmp: Componente name

**Returns** An instance of Entity

**alpha**
Return Entity Transparency. *crafty.entity*

**antigravity**()
Anulates gravity to entity. *crafty.entity*

**Returns** Self, this same entity

**attach**(*entity*)
Attach an entity to this one. *crafty.entity*

**Param** entity: The entity to be attached

**Returns** Self, this same entity

**collision**(**points*)
Constructor takes a polygon or array of points to use as the hit area. *crafty.entity*

The hit area (polygon) must be a convex shape and not concave for the collision detection to work.

Points are relative to the object's position and its unrotated state.

If no parameter is passed, the x, y, w, h properties of the entity will be used, and the hitbox will be resized when the entity is.

If a hitbox is set that is outside of the bounds of the entity itself, there will be a small performance penalty as it is tracked separately.

**Example** ..code:: python

Crafty().e("2D, Collision").collision([50,0], [100,100], [0,100])

**Events**

**NewHitbox [Data: Crafty.polygon]** when a new hitbox is assigned

**Param** *points: Array with an x and y position to generate a polygon

**Returns** Self, this same entity

**color**(*col*)
Creates an entity. *crafty.entity*

**Param** col: new color of the entity

**Returns** Self, this same entity

**detach**(*entity=None*)
Detach an entity from this one. *crafty.entity*

**Param** entity: The entity to be detached, all entities if blank

**Returns** Self, this same entity

**disableControl**()
Disable the component to listen to key events. *crafty.entity*

**Returns** Self, this same entity

**enableControl**()
Enable the component to listen to key events. *crafty.entity*

**Returns** Self, this same entity

**entity**
Entity property. *crafty.entity*

**flip**(*direction*)
Flip entity on passed direction

**Parameters** **direction** – Flip direction

**Returns** Self, this same entity

**fourway**(*speed*)
Creates an four way entity control. *crafty.entity*

**Param** speed: the speed of movement

**Returns** Self, this same entity

**gravity**(*entity*)
Creates gravity to entity. *crafty.entity*

**Param** entity: entity to gravitate to

**Returns** Self, this same entity

**gravityConst**(*g*)
Set gravity to constant g. *crafty.entity*

**Param** g: The gravity constant

**Returns** Self, this same entity

**hit**(*component*)
Takes an argument for a component to test collision for. *crafty.entity*

If a collision is found, an array of every object in collision along with the amount of overlap is passed.

If no collision, will return false. The return collision data will be an Array of Objects with the type of collision used, the object collided and if the type used was SAT (a polygon was used as the hitbox) then an amount of overlap.

```
[{
   obj: [entity],
   type: "MBR" or "SAT",
   overlap: [number]
}]
```

MBR is your standard axis aligned rectangle intersection (.intersect in the 2D component). SAT is collision between any convex polygon.

**Param** component: Check collision with entities that has this component

**Returns** False if no collision. If a collision is detected, returns an Array of objects that are colliding.

**image**(*url*, *repeat='*')
Create a rectangle polygon based on the x, y, w, h dimensions. *crafty.entity*

Draw specified image. Repeat follows CSS syntax ("no-repeat", "repeat", "repeat-x", "repeat-y");

Note: Default repeat is no-repeat which is different to standard DOM (which is repeat)

If the width and height are 0 and repeat is set to no-repeat the width and height will automatically assume that of the image. This is an easy way to create an image without needing sprites.

**Example**

Will default to no-repeat. Entity width and height will be set to the images width and height

..code:: python

ent = Crafty().e("2D, DOM, Image").image("myimage.png")

Create a repeating background.

..code:: python

**bg = Crafty().e("2D, DOM, Image")** .attr(w= Crafty.viewport.width, h= Crafty.viewport.height) .image("bg.png", "repeat");

**Events**

**Invalidate** when the image is loaded

**Param** url: URL of the image.

**Param** repeat: If the image should be repeated to fill the entity.

**Returns** Self, this same entity

**init**()
Create a rectangle polygon based on the x, y, w, h dimensions. *crafty.entity*

By default, the collision hitbox will match the dimensions (x, y, w, h) and rotation of the object.

**Returns** Self, this same entity

**move**(*direction*, *by=1*)
Quick method to move the entity by an amount of pixels. *crafty.entity* in a direction (n, s, e, w, ne, nw, se, sw).

**Parameters**

- **direction** – Direction to move (n,s,e,w,ne,nw,se,sw)
- **by** – Amount to move in the specified direction

**Returns** Self, this same entity

**multiway**(*speed*, ***directions*)
Creates an four way entity control. *crafty.entity*

**Param** speed: the speed of movement

**Param** directions: named directions and degree (UP_ARROW: -90, DOWN_ARROW: 90, RIGHT_ARROW: 0, LEFT_ARROW: 180)

**Returns** Self, this same entity

**onHit**(*component*, *hit*, *nohit=<function <lambda>>*)
Creates an EnterFrame event calling .hit() each frame. *crafty.entity*

When a collision is detected the callback will be invoked.

**Param** hit: Callback method to execute upon collision with component. Will be passed the results of the collision check in the same format documented for hit().

**Param** nohit: Callback method executed once as soon as collision stops.

**Returns** Self, this same entity

**origin**(*value*)
Set rotation origin for entity. *crafty.entity*

**Param** value: lef, top, right, bottom, center, middle

**Returns** Self, this same entity

**rotation**
Rotate entity. *crafty.entity*

**Returns** Ammount of rotation in degrees

**shift**(*x=0*, *y=0*, *w=0*, *h=0*)
Shift or move the entity by an amount. Use negative values for an opposite direction. *crafty.entity*

**Parameters**

- **x** – Amount to move X
- **y** – Amount to move Y
- **w** – Amount to widen
- **h** – Amount to increase height

**Returns** Self, this same entity

**speed**(*speed*)
Change the speed that the entity moves with. *crafty.entity*

**Param** speed: the speed of movement

**Returns** Self, this same entity

**tint**(*color*, *strength*)
Similar to Color by adding an overlay of semi-transparent color. *crafty.entity*

Modify the color and level opacity to give a tint on the entity.

**Example**

..code:: python

Crafty().e("2D, Canvas, Tint").tint("#969696", 0.3)

**Events**

**Invalidate** when the tint is applied

**Param** color: The color in hexadecimal.

**Param** strength: Level of opacity.

**Returns** Self, this same entity

**twoway**(*speed*, *jump=None*)
Creates an two way entity control. *crafty.entity* Constructor to initialize the speed and power of jump. Component will listen for key events and move the entity appropriately. This includes Up Arrow, Right Arrow, Left Arrow as well as W, A, D. Used with the gravity component to simulate jumping.

The key presses will move the entity in that direction by the speed passed in the argument. Pressing the Up Arrow or W will cause the entity to jump.

**Param** speed: the speed of movement

**Param** jump: the speed of jump

**Returns** Self, this same entity

**unflip**(*direction*)
Unflip entity on passed direction (if it's flipped)

**Parameters** **direction** – Unflip direction

**Returns** Self, this same entity

**visible**
Return Entity Visibility. *crafty.entity*

## 6.4 Graphics API

### 6.4.1 Graphic handling classes

**Author** *Carlo E. T. Oliveira*

**Contact** carlo@nce.ufrj.br

**Date** 2014/09/17

**Status** This is a "work in progress"

**Revision** 0.1.0

**Home** Labase

**Copyright** 2013, GPL.

**class** crafty.graphics.**Canvas**(*stage*, *cmp*)
Canvas. *Canvas*

When this component is added to an entity it will be drawn to the global canvas element. The canvas element (and hence all Canvas entities) is always rendered below any DOM entities.

Crafty.canvas.init() will be automatically called if it is not called already to initialize the canvas element.

**draw**(*ctx*, *x*, *y*, *w*, *h*)
draw([[Context ctx, ]Number x, Number y, Number w, Number h]).

**class** crafty.graphics.**Draggable**(*ent*)
Enable drag and drop of the entity. draggable

**disableDrag**()
Stops entity from being draggable. Reenable with .enableDrag().

**dragDirection**(*degrees=None*, *x=None*, *y=None*)
Specify the dragging direction.

if no parameters are given, remove dragging.

**Parameters**

- **degrees** – A number, the degree (clockwise) of the move direction with respect to the x axis.
- **x** – the vector (valx, valy) denotes the move direction.
- **y** – the vector (valx, valy) denotes the move direction.

**enableDrag**()
Rebind the mouse events. Use if .disableDrag has been called.

**startDrag**()
Make the entity follow the mouse positions.

**stopDrag**()
Stop the entity from dragging. Essentially reproducing the drop.

**class** crafty.graphics.**Sprite**(*ent*)
Sprite. *Sprite*

Component for using tiles in a sprite map.

**animate**(*reelId=None*, *loopCount=1*)
Animate Entity.

**Parameters**

- **reelId** – String reel identification
- **loopCount** – Integer number of loops, default 1, indefinite if -1

**Returns** Self, this same entity

**coord**
The coordinate of the slide within the sprite in the format of [x, y, w, h].

**crop**(*x*, *y*, *w*, *h*)
Crop the sprite.

If the entity needs to be smaller than the tile size, use this method to crop it.

The values should be in pixels rather than tiles.

**Parameters**

- **x** – Offset x position
- **y** – Offset y position
- **w** – New width
- **h** – New height

**Returns** Self, this same entity

**isPlaying**(*reelId=''*)
Return is the reel is playing.

**Parameters** **reelId** – The reelId of the reel we wish to examine, if missing default to current reel

**Returns** The current animation state

**loops**(*loopCount=None*)
Set or return the number of loops.

Sets the number of times the animation will loop for. If called while an animation is in progress, the current state will be considered the first loop.

**Parameters** **loopCount** – The number of times to play the animation, if missig retun loops left.

**Returns** The number of loops left. Returns 0 if no reel is active.

**pauseAnimation**()
Pauses the currently playing animation, or does nothing if no animation is playing.

**reel**(*reelId*, *duration*, *fromX*, *fromY*, *frameCount*)
Create animation reel.

**Param** String reelId, Duration duration, Number fromX, Number fromY, Number frameCount

**Returns** Self, this same entity

**reelPosition**(*position=None*)
Sets the position of the current reel by frame number.

**Parameters** **position** – The frame to jump to. This is zero-indexed. A negative values counts back from the last frame. Sets the position of the current reel by percent progress if number is float. Jumps to the specified position if string. The only currently accepted value is "end", which will jump to the end of the reel.

**Returns** The current frame number

**resetAnimation**()
Resets the current animation to its initial state.

Resets the number of loops to the last specified value, which defaults to 1.

Neither pauses nor resumes the current animation.

**resumeAnimation**()
This will resume animation of the current reel from its current state.

If a reel is already playing, or there is no current reel, there will be no effect.

**sprite**(*x*, *y*, *w*, *h*)

**tween**(*duration*, ***properties*)
This method will animate numeric properties over the specified duration.

These include x, y, w, h, alpha and rotation in degrees.

**Parameters**

- **properties** – Object of numeric properties and what they should animate to
- **duration** – Duration to animate the properties over, in milliseconds.

**Returns** The current frame number

CHAPTER 7

# Indices and tables

- genindex
- modindex
- search

## C

crafty.base *(Web)*, 13
crafty.core *(Web)*, 18
crafty.entity *(Web)*, 20
crafty.graphics *(Web)*, 24

## A

alpha (crafty.entity.Entity attribute), 20
animate() (crafty.graphics.Sprite method), 25
antigravity() (crafty.entity.Entity method), 20
attach() (crafty.entity.Entity method), 20
attr() (crafty.base.Base method), 13

## B

background() (crafty.base.Base method), 13
Base (class in crafty.base), 13
BCrafty (class in crafty.core), 18
bind() (crafty.base.Base method), 13
body (crafty.core.document attribute), 20
bounds() (crafty.base.ViewPort method), 15

## C

c() (crafty.core.BCrafty method), 18
Canvas (class in crafty.graphics), 25
canvas() (crafty.core.BCrafty method), 18
centerOn() (crafty.base.ViewPort method), 16
clampToEntities (crafty.base.ViewPort attribute), 16
collision() (crafty.entity.Entity method), 20
color() (crafty.entity.Entity method), 21
coord (crafty.graphics.Sprite attribute), 25
crafty (crafty.core.BCrafty attribute), 18
crafty() (crafty.base.Base method), 14
crafty.base (module), 13
crafty.core (module), 18
crafty.entity (module), 20
crafty.graphics (module), 24
crop() (crafty.graphics.Sprite method), 25

## D

destroy() (crafty.base.Base method), 14
detach() (crafty.entity.Entity method), 21
disableControl() (crafty.entity.Entity method), 21
disableDrag() (crafty.graphics.Draggable method), 25
document (class in crafty.core), 20
dragDirection() (crafty.graphics.Draggable method), 25
Draggable (class in crafty.graphics), 25
draw() (crafty.graphics.Canvas method), 25

## E

e() (crafty.core.BCrafty method), 18
enableControl() (crafty.entity.Entity method), 21
enableDrag() (crafty.graphics.Draggable method), 25
Entity (class in crafty.entity), 20
entity (crafty.entity.Entity attribute), 21

## F

flip() (crafty.entity.Entity method), 21
follow() (crafty.base.ViewPort method), 16
fourway() (crafty.entity.Entity method), 21

## G

gravity() (crafty.entity.Entity method), 21
gravityConst() (crafty.entity.Entity method), 21

## H

hit() (crafty.entity.Entity method), 22

## I

image() (crafty.entity.Entity method), 22
init() (crafty.base.ViewPort method), 16
init() (crafty.entity.Entity method), 22
isDown() (crafty.base.Base method), 14
isPlaying() (crafty.graphics.Sprite method), 26

## K

keys (crafty.base.Base attribute), 14

## L

load() (crafty.core.BCrafty method), 19
loops() (crafty.graphics.Sprite method), 26

## M

mouselook() (crafty.base.ViewPort method), 16
mousePos (crafty.base.Base attribute), 14
move() (crafty.entity.Entity method), 23

multiway() (crafty.entity.Entity method), 23

## O

onebind() (crafty.base.Base method), 14
onHit() (crafty.entity.Entity method), 23
origin() (crafty.entity.Entity method), 23

## P

pan() (crafty.base.ViewPort method), 16
pauseAnimation() (crafty.graphics.Sprite method), 26

## R

randRange() (crafty.core.BCrafty method), 19
reel() (crafty.graphics.Sprite method), 26
reelPosition() (crafty.graphics.Sprite method), 26
resetAnimation() (crafty.graphics.Sprite method), 26
resumeAnimation() (crafty.graphics.Sprite method), 26
rotation (crafty.entity.Entity attribute), 23

## S

scale() (crafty.base.ViewPort method), 17
scene() (crafty.core.BCrafty method), 19
scroll() (crafty.base.ViewPort method), 17
shift() (crafty.entity.Entity method), 23
speed() (crafty.entity.Entity method), 23
Sprite (class in crafty.graphics), 25
sprite() (crafty.core.BCrafty method), 19
sprite() (crafty.graphics.Sprite method), 27
spritem() (crafty.core.BCrafty method), 19
sprites() (crafty.core.BCrafty method), 19
startDrag() (crafty.graphics.Draggable method), 25
stopDrag() (crafty.graphics.Draggable method), 25

## T

text() (crafty.base.Base method), 14
textColor() (crafty.base.Base method), 15
textFont() (crafty.base.Base method), 15
tint() (crafty.entity.Entity method), 24
tween() (crafty.graphics.Sprite method), 27
twoway() (crafty.entity.Entity method), 24

## U

unbind() (crafty.base.Base method), 15
unflip() (crafty.entity.Entity method), 24
unselectable() (crafty.base.Base method), 15

## V

ViewPort (class in crafty.base), 15
visible (crafty.entity.Entity attribute), 24

## X

x (crafty.base.Base attribute), 15
x (crafty.base.ViewPort attribute), 17

## Y

y (crafty.base.Base attribute), 15
y (crafty.base.ViewPort attribute), 17

## Z

zoom() (crafty.base.ViewPort method), 17