cocos.actions.move_actions module

Actions for moving things around based on their velocity and acceleration.

The simplest usage:

```
sprite = cocos.sprite.Sprite('ship.png') sprite.velocity = (100, 100)
sprite.do(Move())
```

This will move the sprite (100, 100) pixels per second indefinitely.

Typically the sprite would be controlled by the user, so something like:

```
keys = <standard pyglet keyboard state handler>

class MoveShip(Move):
   def step(self, dt):
       super(MoveShip, self).step(dt)
       self.target.dr = (keys[key.RIGHT] - keys[key.LEFT]) * 360
       rotation = math.pi * self.target.rotation / 180.0
       rotation_x = math.cos(-rotation)
       rotation_y = math.sin(-rotation)
       if keys[key.UP]:
           self.target.acceleration = (200 * rotation_x, 200 * rotation_y)

ship.do(MoveShip())
```

class **Move**(**args*, ***kwargs*)

Bases: **cocos.actions.base_actions.Action**

Move the target based on parameters on the target.

For movement the parameters are:

```
target.position = (x, y)
target.velocity = (dx, dy)
target.acceleration = (ddx, ddy) = (0, 0)
target.gravity = 0
```

And rotation:

```
target.rotation
target.dr
target.ddr
```

step(*dt*)

class **WrappedMove**(**args*, ***kwargs*)

Bases: **cocos.actions.move_actions.Move**

Move the target but wrap position when it hits certain bounds.

Wrap occurs outside of 0 < x < width and 0 < y < height taking into account the

dimenstions of the target.

init(*width*, *height*)

Init method.

Parameters:

width : *integer*

The width to wrap position at.

height : *integer*

The height to wrap position at.

step(*dt*)

class **BoundedMove**(**args*, ***kwargs*)

Bases: **cocos.actions.move_actions.Move**

Move the target but limit position when it hits certain bounds.

Position is bounded to 0 < x < width and 0 < y < height taking into account the dimenstions of the target.

init(*width*, *height*)

Init method.

Parameters:

width : *integer*

The width to bound position at.

height : *integer*

The height to bound position at.

step(*dt*)

class **Driver**(**args*, ***kwargs*)

Bases: **cocos.actions.base_actions.Action**

Drive a *CocosNode* object around like a car in x, y according to a direction and speed.

Example:

```
# control the movement of the given sprite
sprite.do(Driver())

...
sprite.rotation = 45
sprite.speed = 100
...
```

The sprite MAY have these parameters (beyond the standard position and rotation):

speed : *float*

Speed to move at in pixels per second in the direction of the target's rotation.

acceleration : *float*

If specified will automatically be added to speed. Specified in pixels per second per second.

max_forward_speed : *float (default None)*

Limits to apply to speed when updating with acceleration.

max_reverse_speed : *float (default None)*

Limits to apply to speed when updating with acceleration.

`step`(*dt*)