wasabi.physics Documentation

Release 0.1.1

Daniel Pope

August 19, 2016

Contents

wasabi.physics is a physics simulation intended for creating 2D platform games.

Contents:

CHAPTER 1

Physics Simulation

wasabi.physics.Physics is physical simulation which given a (small) time step will update the position and velocity vectors of a number of *Bodies* that have been registered with the simulation.

The basic pattern for using the physics engine is as follows:

1. Set up the *Physics* with all the *dynamic bodies* and *static geometry* that will be interacting in the world.

Then, each 'frame':

1. Apply any forces that are acting on the bodies.
2. Call `Phsyics.update()` to update the simulation.
3. Retrieve the updated positions of the bodies.

1.1 Simulation

class `wasabi.physics.`**Physics**

add_body(*body*)
Add a dynamic body to the world.

Parameters **body** – The Body to add.

add_static(*body*)
Add static geometry to the world.

Parameters **body** – The StaticBody to add.

ray_query(*segment*, *mask=65535*)
Query dynamic and static bodies that intersect the given `Segment`.

The return value is a list of tuples (distance, object), sorted in ascending order of distance. Each tuple represents an intersection with a new object.

For each tuple, *distance* is the distance of the point of first intersection of the object along the segment.

If the hit is with a dynamic object then *object* will be the Body that was hit.

If the hit is with static geometry then *object* will be StaticBody (the class, not an instance).

Parameters

- **segment** – The segment to query.

- **mask** – A collision bitmask. When set, only dynamic objects whose group bitmask ANDed with the mask bitmask will be returned. All static objects are returned.

remove_body(*body*)
Remove a Body from the world.

Parameters **body** – The Body to remove.

remove_static(*body*)
Remove static geometry from the world.

Parameters **body** – The StaticBody to remove.

update(*dt*)
Update all dynamic objects added to this Physics system.

Parameters **dt** – The amount of time (in seconds) to advance the simulation.

1.2 Bodies

Bodies are the objects that can be added to the simulation. These may represent controllable players or simply movable scenery.

class wasabi.physics.**Body**(*rect*, *mass*, *pos=Vector((0, 0))*, *controller=None*, *groups=1*, *mask=255*)
A dynamic rectangle whose velocity, position are controlled by Physics.

Bodies should be manipulated by applying forces, etc.

Each body has mass, which causes inertia.

pos
The position vector of the body. This is updated by the simulation.

It can be updated directly, although it is preferable to apply forces and let ths simulation update the body's position.

v
The velocity vector of the body. This is updated by the simulation.

Like pos, it can also be updated directly, but it is better to manipulate it using `Body.add_impulse()`.

rect
The shape of the body (a `Rect <wasabi.geom.poly.Rect>`), disregarding its current position. This can be updated, for example in the case of a body that grows or shrinks.

groups
A bitvector defining the collision groups of which this body is a member.

mask
A bitvector defining the collision groups with which this body will collide.

apply_force(*f*)
Apply a force to the body for the next update of the simulation.

Parameters **f** – The force vector to apply.

apply_impulse(*impulse*)
Apply an impulse to the body.

This causes an instant change in the velocity of the body.

Parameters **impulse** – The impulse vector to apply.

get_rect()
Get the current rectangle of the body.

reset_forces()
Reset the forces acting on the body.

Subclasses can override this method to apply forces that apply constantly (gravity, wind, bouyancy etc).

set_collision_handler(*handler*)
Set a callback to be called when this body collides with another.

The callback must accept these arguments:

```
handler(b, dt)
```

where *b* is the other Body involved in the collision, and *dt* is the simulation's time step.

class wasabi.physics.**FloatingBody**(*rect*, *mass*, *pos=Vector((0, 0))*, *controller=None*, *groups=1*, *mask=255*)
A dynamic body on which gravity does not apply.

class wasabi.physics.**StaticBody**(*rectangles*, *pos=Vector((0, 0))*)
A static body represents an immovable object.

Static bodies should be added to the Physics with *Physics.add_static()* rather than *.add_body*.

1.3 Collision Groups and Masks

To allow for types of objects that don't collide with every other type of object, each *Body* object can be configured with a bitvector that defines its collision group, and a bitvector that defines the groups with which it will collide.

For example, if we define collision groups as follows:

```
FOREGROUND = 1
BACKGROUND = 1 << 2
```

Then we could create a body that only exists in the background by setting its `groups` to `BACKGROUND`. If it should collide only with other background objects we should set its `mask` to `BACKGROUND` also.

Similarly we could create an body that will only collide with foreground objects by setting `mask` to `FOREGROUND`, or we could | (or) together the groups to create an object that will collide with both groups:

```
mask=FOREGROUND | BACKGROUND
```

Note: It is possible for one object to collide with another object whose groups do not match its mask, because the condition may be true for the other object.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

W