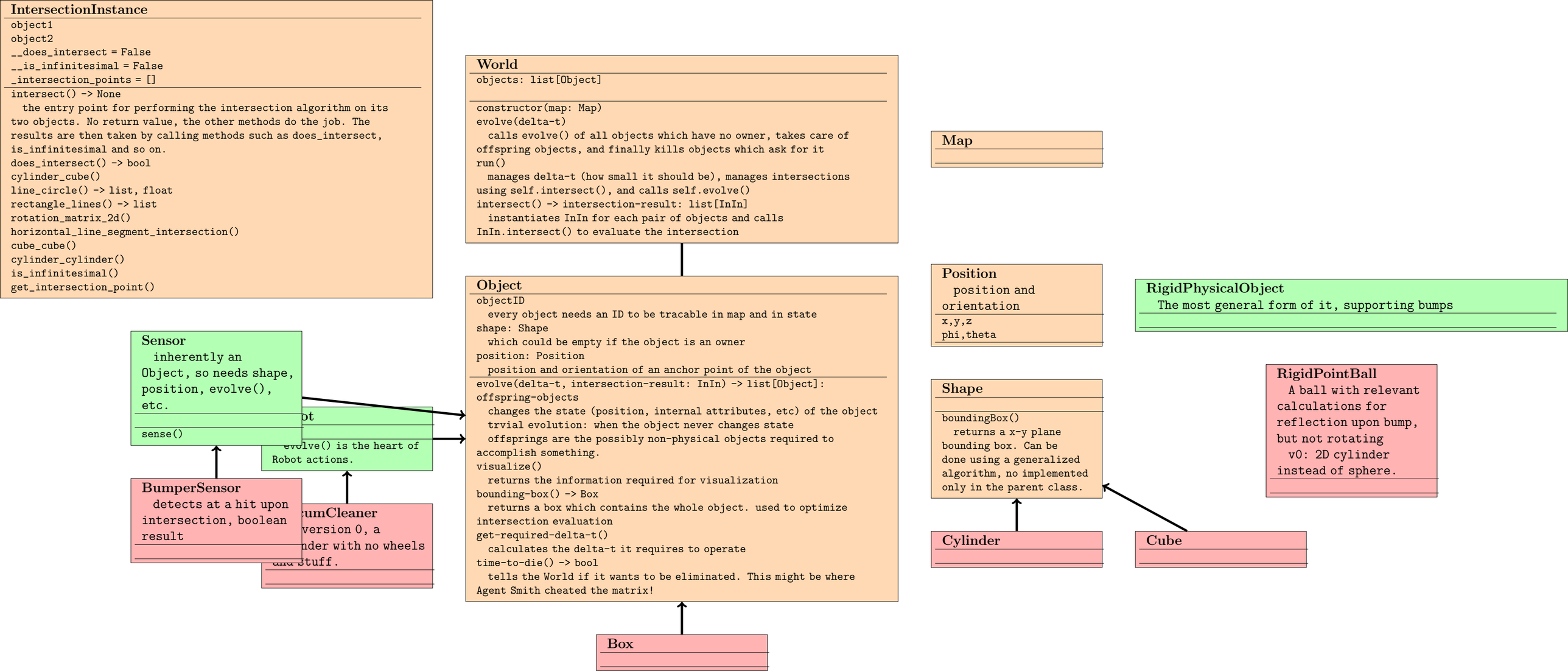
IntersectionInstance
object1
object2
__does_intersect = False
__is_infinitesimal = False
_intersection_points = []
intersect() -> None
the entry point for performing the intersection algorithm on its two objects. No return value, the other methods do the job. The results are then taken by calling methods such as does_intersect, is_infinitesimal and so on.
does_intersect() -> bool
cylinder_cube()
line_circle() -> list, float
rectangle_lines() -> list
rotation_matrix_2d()
horizontal_line_segment_intersection()
cube_cube()
cylinder_cylinder()
is_infinitesimal()
get_intersection_point()
World
objects: list[Object]
constructor(map: Map)
evolve(delta-t)
calls evolve() of all objects which have no owner, takes care of offspring objects, and finally kills objects which ask for it
run()
manages delta-t (how small it should be), manages intersections using self.intersect(), and calls self.evolve()
intersect() -> intersection-result: list[InIn]
instantiates InIn for each pair of objects and calls InIn.intersect() to evaluate the intersection
Map
Object
objectID
every object needs an ID to be tracable in map and in state
shape: Shape
which could be empty if the object is an owner
position: Position
position and orientation of an anchor point of the object
evolve(delta-t, intersection-result: InIn) -> list[Object]:
offspring-objects
changes the state (position, internal attributes, etc) of the object
trvial evolution: when the object never changes state
offsprings are the possibly non-physical objects required to accomplish something.
visualize()
returns the information required for visualization
bounding-box() -> Box
returns a box which contains the whole object. used to optimize intersection evaluation
get-required-delta-t()
calculates the delta-t it requires to operate
time-to-die() -> bool
tells the World if it wants to be eliminated. This might be where Agent Smith cheated the matrix!
Position
position and orientation
x,y,z
phi,theta
RigidPhysicalObject
The most general form of it, supporting bumps
Sensor
inherently an Object, so needs shape, position, evolve(), etc.
sense()
ot
evolve() is the heart of Robot actions.
BumperSensor
detects at a hit upon intersection, boolean result
umCleaner
version 0, a
nder with no wheels
and stuff.
Shape
boundingBox()
returns a x-y plane bounding box. Can be done using a generalized algorithm, no implemented only in the parent class.
RigidPointBall
A ball with relevant calculations for reflection upon bump, but not rotating
v0: 2D cylinder instead of sphere.
Cylinder
Cube
Box