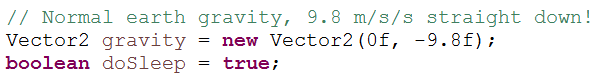
**Box2D – HamuDEV – Covisoft**

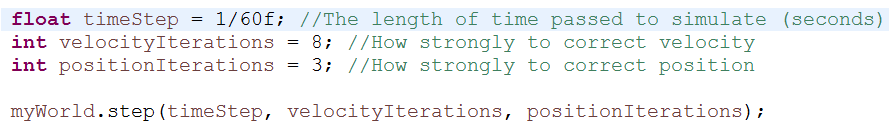
1. **World settings**

* Worlds were briefly mentioned in one of the earlier topics as being the main entity in which all the Box2D bodies live. When you create or delete a body, you call a function of the world object to do this, so the world is managing all the allocations for the objects within it too. This means that the world is pretty important, so let's take a look at what we can do with one.
  + define gravity *(Trọng lực)*
  + tune the physics simulation
  + find fixtures in a given region
  + cast a ray and find intersected fixtures
* A world is set up like any normal class, taking a couple of fundamental settings in the constructor.



The sleep parameter says whether bodies should be allowed to 'sleep' if nothing is happening to them, for efficiency. If this is set to true, bodies will sleep when they come to rest, and are excluded from the simulation until something happens to 'wake' them again.

Once you have a world created as above, you can add bodies into it as we've been doing. To make anything interesting happen, we need to repeatedly call the Step function of the world to run the physics simulation. This is also being handled by the testbed framework, as part of the Step function of the Test class.



The velocity iterations and position iterations settings affect the way bodies will react when they collide. Typically in Box2D when a collision between two objects is detected, those objects are overlapping (stuck into other) and some calculation needs to be done to figure out how each body should move or rotate so that they are not overlapping any more. Making these values higher will give you a more correct simulation, at the cost of some performance.

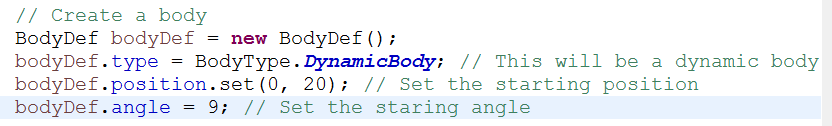
* Cleaning up



1. **Bodies**

* Bodies are the fundamental objects in the physics scene, but they are not what you actually see bouncing around and colliding with each other.
* You can think of a body as the properties of an object that you cannot see (draw) or touch (collide with). These invisible properties are:
* **mass** - how heavy it is *(Khối lượng)*
* **velocity** - how fast and which direction it's moving *(Vận tốc)*
* **rotational** **inertia** - how much effort it takes to start or stop spinning *(Quán tính)*
* **angular** **velocity** - how fast and which way it's rotating *(Tốc độ quay tròn)*
* **location** - where it is *(Vị trí)*
* **angle** - which way it is facing *(Góc quay)*
* **Creating a body**

Bodies are made by first setting up a definition, and then using this to create the body object itself. This can be handy if you want to make many bodies which are all the same, or very similar. Add the following code to set up a definition for a body:



That's enough to define a basic body definition. Remember a body does not have any size, shape, so we don't define those here. You may be wondering why it has no mass yet - the usual way of providing a mass for a body is by adding fixtures to it, which is coming up in the next step. Now, use this definition to create the actual body instance:



* **Static bodies**



Now let's see what a static body does. Since we already have definitions for a body and a fixture, we can re-use them and just change the necessary features.

* **Kinematic bodies**



Lastly, let's see what a 'kinematic' body is all about. As we have seen so far, dynamic bodies move and static bodies don't. When a static body and a dynamic body collide, the static body always 'wins' and holds its ground, and the dynamic body will retreat as necessary so that the two are not overlapping. A kinematic body is very similar to a static body in that when it collides with a dynamic body it always holds its ground and forces the dynamic body to retreat out of the way. The difference is that a kinematic body can move.

* **Iterating over the bodies in the world**

If you want to look at all the bodies in the world, you can do it as below. The function GetBodies() returns the first element in a linked list of bodies.



1. **Fixtures**

* Fixtures are used to describe the size, shape, and material properties of an object in the physics scene. One body can have multiple fixtures attached to it, and the center of mass of the body will be affected by the arrangement of its fixtures. When two bodies collide, their fixtures are used to decide how they will react. The main properties of fixtures are:
  + **shape** - a polygon or circle
  + **restitution** - how bouncy the fixture is *(Phản lực)*
  + **friction** - how slippery it is *(Độ ma sát)*
  + **density** - how heavy it is in relation to its area *(Độ dày)*
* **Shape**

Every fixture has a shape which is used to check for collisions with other fixtures as it moves around the scene. A shape can be a circle or a polygon. Let's set up a circle shape...

* **Density**
* **Multiple fixtures**
* **Friction**
* **Restitution**

1. **World settings**