API Reference

```
public static void WaterCutter.CookCache(Mesh m, ref tri[] _triangles, ref tri[] worldBuffer,
                                         ref tri[] wetTris, ref tri[] dryTris);
```

This function allocate memory for the cache, and setup mesh cache. It must be called first, usually in the Awake() function.

```
Mesh m                  The mesh used to compute buoyancy forces.
ref tri[] _triangles    An array of triangles  generated by the function. You must keep it.
ref tri[] worldBuffer   An working array allocated by this function. Must be kept.
ref tri[] wetTris       This array will store submerged triangles. Must be kept.
ref tri[] dryTris       This array will store unsubmerged triangles. Must be kept.
```

```
public static void WaterCutter.CookMesh(Vector3 p, Quaternion r, ref tri[] _triangles,
                                        ref tri[] worldBuffer);
```

This function must be called once by simulation frame. It will prepare the mesh cache for this frame. Arrays given to this function must be the same as the one received after executing WaterCutter.CookCache.

```
Vector3 p               Position of the object during the frame (Transform.position)
Quaternion r            Rotation of the object during the frame (Transform.rotation)
ref tri[] _triangles    triangle cache array, obtained using WaterCutter.CookCache
ref tri[] worldBuffer   array obtained using WaterCutter.CookCache
```

```
public static void WaterCutter.SplitMesh(tri[] raw, ref tri[] wet, ref tri[] dry, out uint wetNbr,
                                         out uint dryNbr, WaterSurface.GetWaterHeight fwater);
```

This function will compute submerged triangles, and emerged triangles, splitting them correctly. This function must be called just after WaterCutter.cookMesh.

```
tri[] raw               The worldBuffer array obtained by SplitMesh function.
ref tri[] wet           Output submerged triangles here. You must use cookCache's arrays.
ref tri[] dry           Output unsubmerged triangles here. You must use cookCache's arrays.
out uint wetNbr         Output the number of submerged triangles. Different from wet.Length() !
out uint dryNbr         Output the number of unsubmerged triangles. Different from dry.Length() !
WaterSurface.GetWaterHeight fwater      Delegate used to find water height at a position (X, -, Z)
```

```
public static void Archimeds.ComputeAllForces(tri[] wetTris, tri[] dryTris, uint nbrWet, uint nbrDry,
                                              Vector3 speed, Rigidbody rb)
```

This function is the last one you need to call. Using triangles generated by WaterCutter.SplitMesh, it will compute all the forces and apply them to the Rigidbody **rb.**

```
tri[] wetTris   The array obtained through SplitMesh.
tri[] dryTris   The array obtained through SplitMesh.
uint nbrWet     The number nbrWet obtained through SplitMesh.
uint nbrDry     The number nbrDry obtained through SplitMesh.
Vector3 speed   The current speed of the object. Often rigidbody.speed.
Rigidbody rb    The Rigidbody on which you want to apply the forces.
```

Custom wave generators

To use your own wave generators functions, you can use this delegate.

```
public delegate float GetWaterHeight(Vector3 pos);
```

This function return the absolute Y world position of water surface at a given world position (X,Z).
This function should be fast ! It will be called 3 time per triangles, each frame.

Built in wave generators

```
public static GetWaterHeight simpleWater;       Generate a heavy ocean, used in Sample.unity.
public static GetWaterHeight flatWater;         Generate a flat surface. Used for rivers, pounds, etc…
```