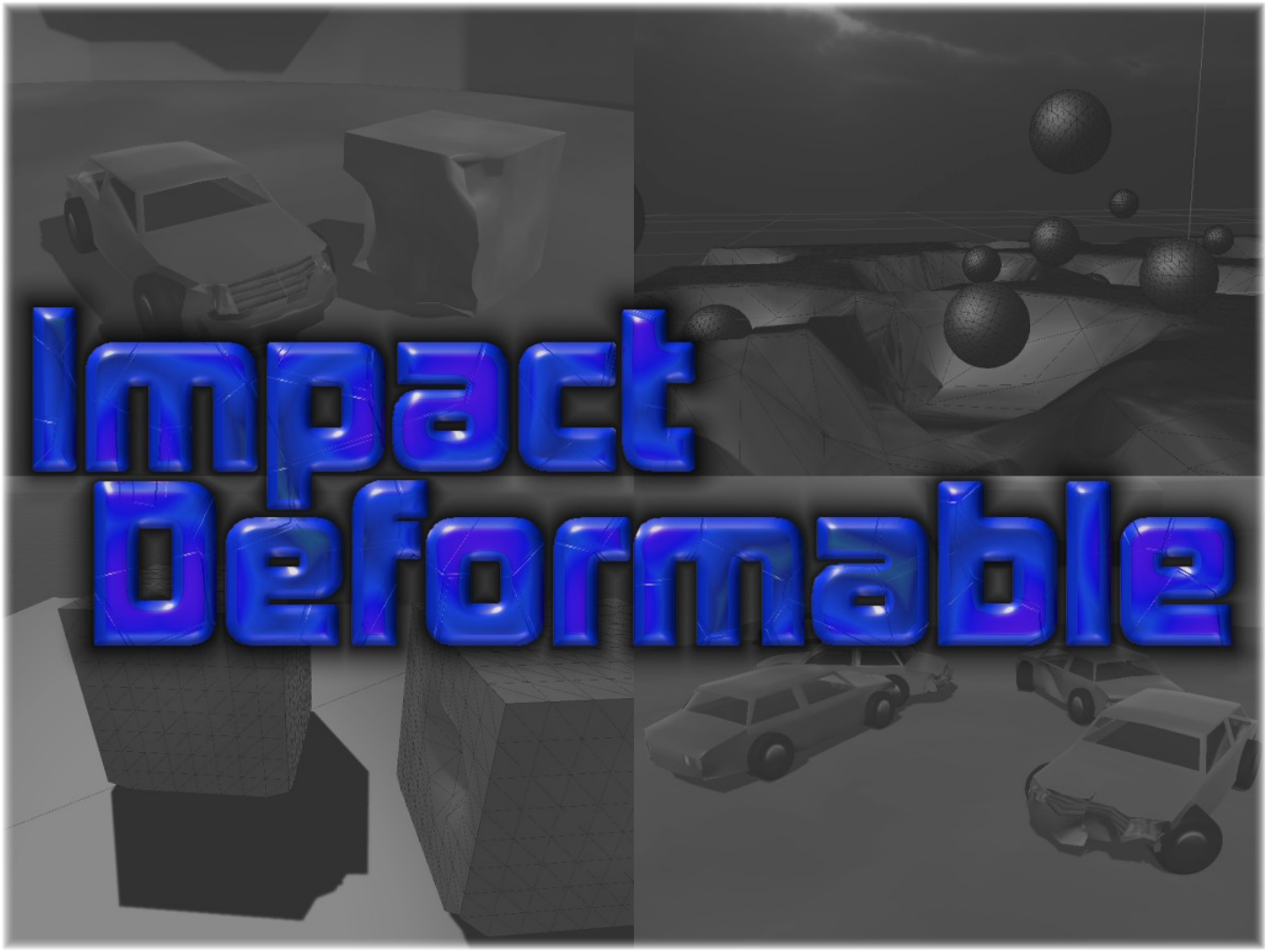


About

Impact deformable was designed to provide automatic mesh deformation from physical impacts.

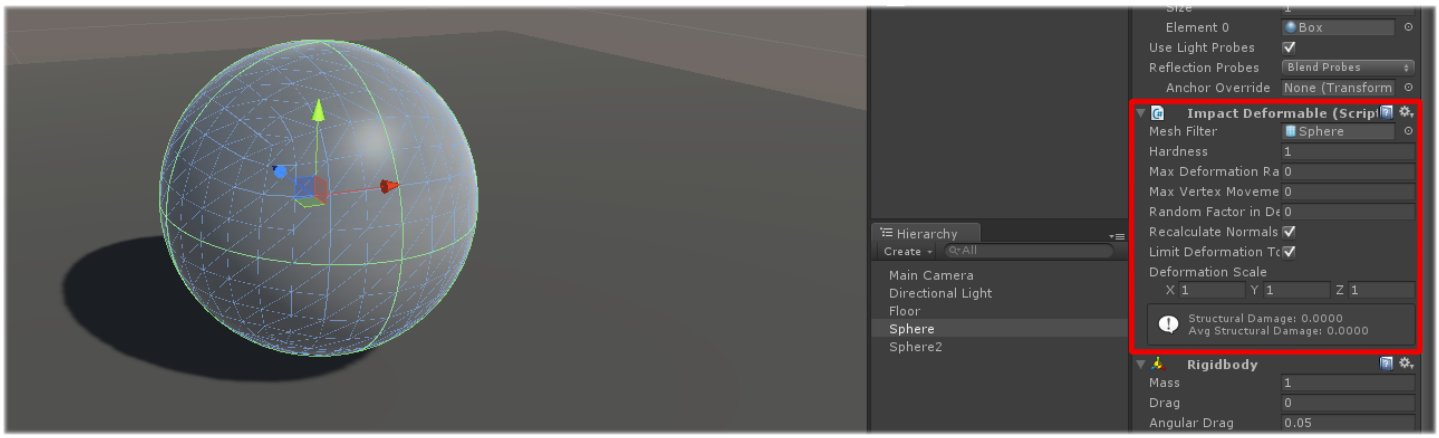
It has been used in many styles of Unity game projects (race/driving, war simulations, real time strategy) targeting all platforms.

This is the second major version with many fixes and optimizations acquired from six years of users feedback.

Quick Start

Attach the ImpactDeformable.cs script to any object with a mesh filter and a 3D collider to make it deformable by impacts.

This is the setting present in the first demo scene (**Simple.unity**) that comes along with the package.



The demo scene Simple.unity, a standard Unity sphere with Impact Deformable script attached.

Not so quick start

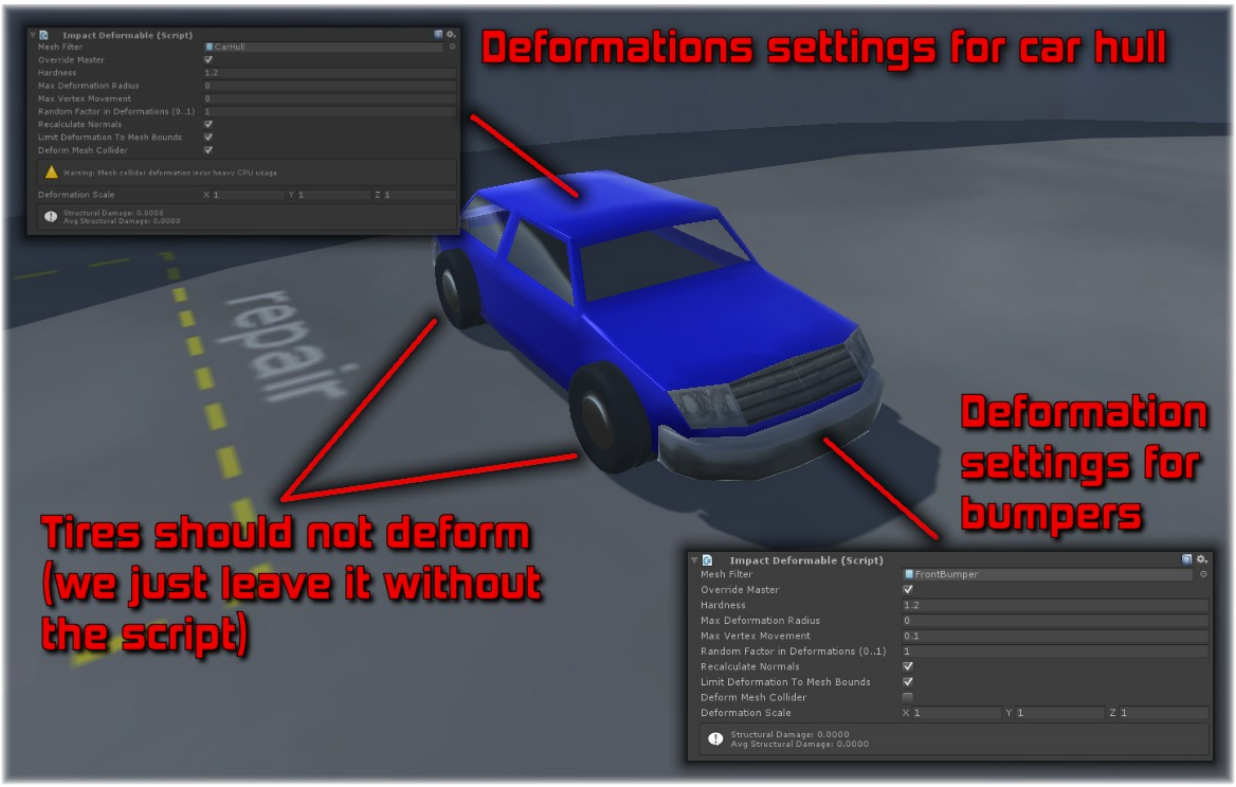
Impact Deformable is designed to be as versatile as possible in order to cover all colliders x meshes configurations that real games might use.

In essence, it transforms OnCollisionEnter and OnCollisionStay events into visible mesh deformations.

By default, it is linked to the MeshFilter in the same game object, but this can be changed with the MeshFilter property - effectively making it receive impacts from one object and deforming another.

The deformation process is applied to existing vertices on the mesh - no new geometry is created, so deformation quality is tied with mesh topology.

Compound Colliders



Compound colliders, with per object deformation settings

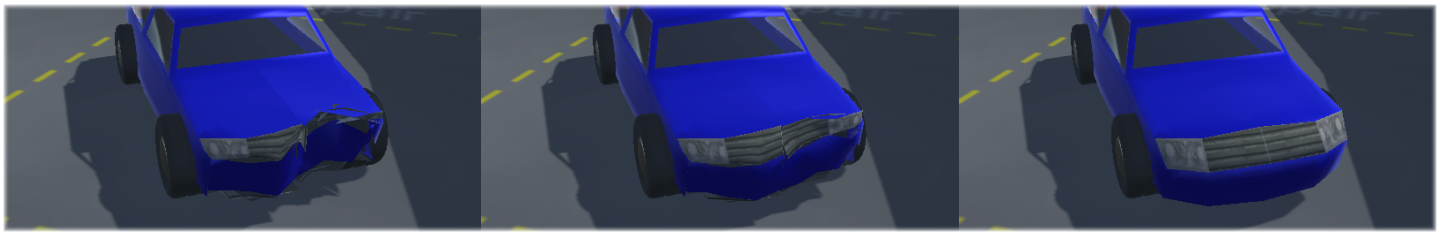
In Unity, it is a common practice to structure a root object with a RigidBody and some children objects with primitive colliders describing a complex shape (aka Compound Colliders).

To perform proper mesh deformation in a compound collider structure, impact deformable switches automatically to hierarchical mode.

This is done under the hood without the need of any user action. All that is left to the user is to attach an Impact Deformable in each sub object with MeshFilter that needs to be deformed. This is necessary because sometimes one needs to specify different settings for each object piece (hardness, for example) or disable deformation for a particular piece (think of car tires).

The demo scene **CarDerby.unity** is an example of using Impact Deformable with compound colliders.

Repairs



Progressive repair, restoring mesh to its original state

Repairs can be performed by a percentage value, either in the whole mesh structure or area delimited.

In both cases the method to be used is the **Repair(percentage, point, radius)**

Point parameter is in world space.

Here is an example for reparing an object area with mouse down:

```
if (Input.GetMouseButton(0))
{
     foreach (Camera cam in Camera.allCameras)
     {
          Ray ray = cam.ScreenPointToRay(Input.mousePosition);
          RaycastHit hit = new RaycastHit();
          if (Physics.Raycast(ray, out hit))
          {
               ImpactDeformable impactDeformable = hit.collider.GetComponent<ImpactDeformable>();
               if (impactDeformable != null)
                    impactDeformable.Repair(0.05f, hit.point, 0.75f);
          }
     }
 }
```

From version 2.1, repair will also revert vertex color deformation (if any) to original color.

Please check the **CarDerby.unity3d** demo scene for more repair examples. In this demo any object can be repaired with mouse down over it (area delimited) and the player car can be repaired (whole structure) if over repair box painted on ground.

Damage estimation

Damage estimation is one of the new features of the version 2 of Impact Deformable. In essence, it measures the average vertex deformation in mesh space.

The idea is to integrate this value into game rules, so game entities "damage" are tied to what players see in game objects.

The damage estimation is requested by reading **StructuralDamage** property. This value is cached internally for good performance. In compound colliders, the property **AverageStructuralDamage** can be used to get average damage among entire structure.

Example:

```
public float CarDamage
{
	get
	{
		return Mathf.Clamp01(CarHullDeformable.StructuralDamage / 0.065f);
	}
}
```

Please check the **CarDerby.unity3d** demo scene for damage estimation example. In this demo the damage estimation is integrated into game rules in 2 ways: First, if a car receives enough damage it will be disabled. Second, bumpers falls when damaged beyond a point. Finally, there is a structure "health" bar measuring player car damage.

A bar showing damage estimation for an impact deformable object.

Manual deformations

Beside automatic deformations by impact one can also request arbitrary deformations to Impact Deformable.

This can be done with a simple call to the public **Deform(point, force)** method. Please note that both point and force parameters are in world space.

For example, to apply random deformation to all Impact Deformable objects in scene:

```
FindObjectsOfType<ImpactDeformable>()
     .ToList()
     .ForEach(i =>
         {
             Collider collider = i.GetComponent<Collider>();
             if (collider == null)
                 return;

             Vector3 v = (Random.onUnitSphere * 5) + i.transform.position;
             Ray ray = new Ray(v, i.transform.position - v);

             RaycastHit hit;
             if (collider.Raycast(ray, out hit, 10))
                 i.Deform(hit.point, ray.direction.normalized * 0.3f);
         });
```

Please check the demo scene **CarDerby.unity3d** where there is a button "Deform all objects" with random manual deformation applied.

Settings (Properties)

MeshFilter

Points to a MeshFilter with the mesh to be deformed. By default it is the MeshFilter attached to the same object where Impact Deformable is (if any).

Hardness

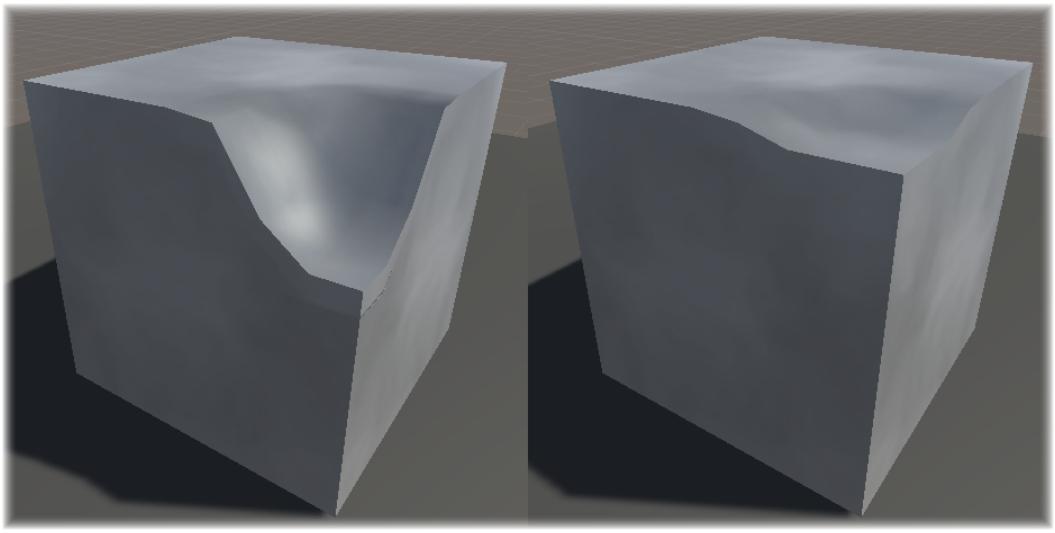
Specifies amount of distortion that this object will suffer from each impact. Higher values make harder objects that are less deformed by collisions.

Default value is 1.

Different hardness settings against same impact force.

MaxVertexMov

Limit the amount that a vertex can move from its original position in deformations (object space).
0 (default value) means no limit.



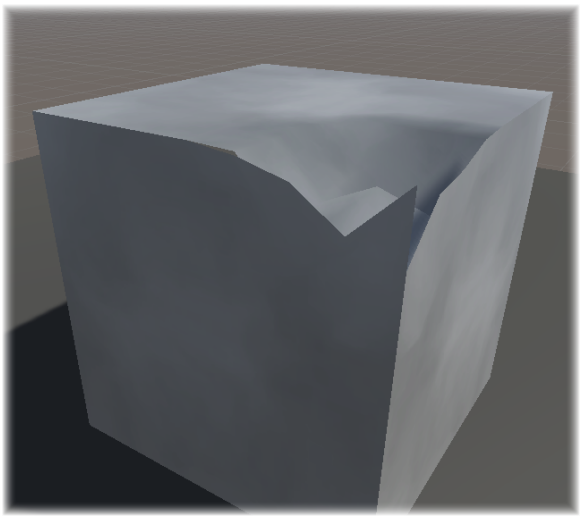
MaxVertexMov effect on deformation (right cube)

MaxDeformationRadius

Impact deformable estimates how many vertices are affected by an impact by the impact force.
This property can limit the maximum deformation radius.
0 (default value) means no limit.

RandomFactorDeformation

If above 0 applies random variations in each vertex deformation, creating a "shattered" effect.
0 (default value) turns it off, 1 is the maximum value.



RandomFactorDeformation effect

DeformationsScale

A vector to scale deformations in any shape (in mesh space).

Impact Deformable was built respecting PhysX/Unity scale of 1 = 1 meter

If this object is out of scale (cars of 30 meters for example), this can be used to scale deformations proportionally

Also, can be used to perform some tricks. (1, 0, 1) for example, would disable the Y component of deformations while (3, 1, 1) would increase X deformation 3 times.

(1, 1, 1) is the default value that applies no deformation scale.

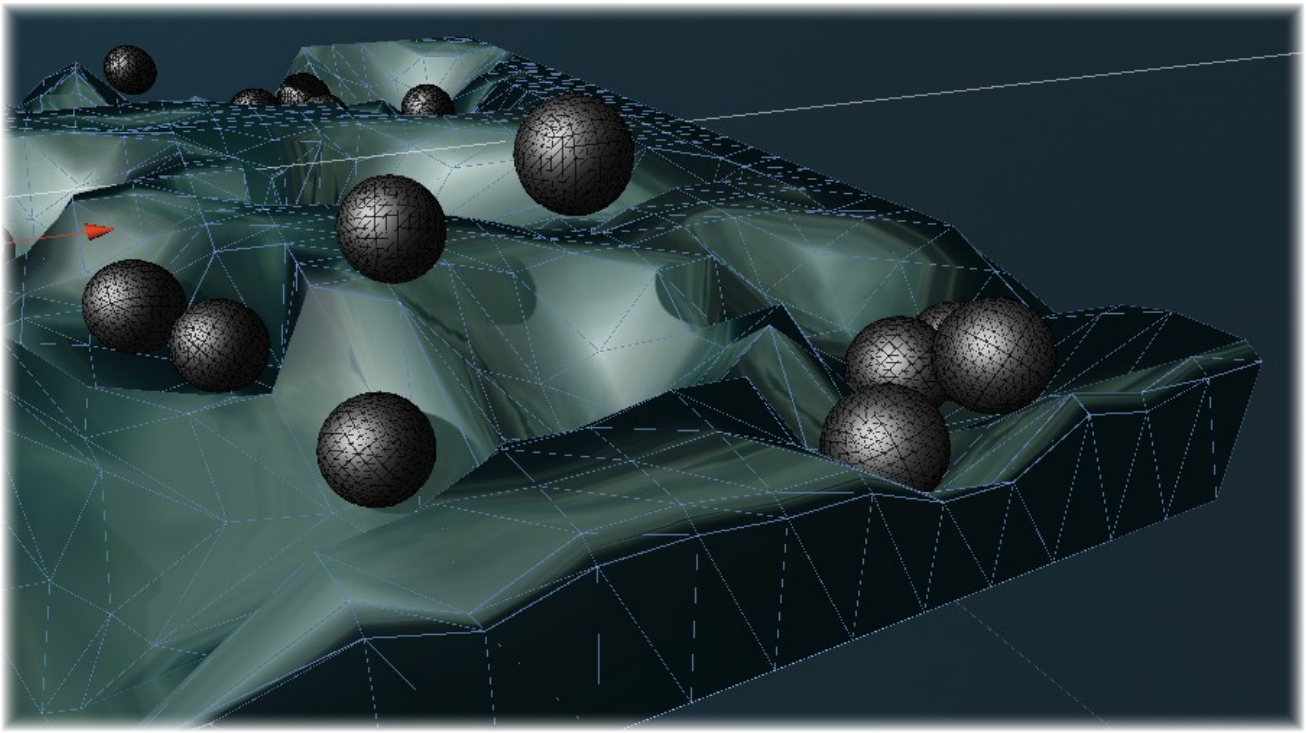
DeformMeshCollider

Impact deformable can be instructed to deform the attached MeshCollider along with the mesh. Both convex and non-convex MeshColliders can be deformed in this way.

This can produce amazing effects like the one seen on the **MetalRain.unity3d** demo scene, where impacts change the collision shape of the floor. There is, however, a high CPU cost involved in the process.

This is only visible in inspector if there is a MeshCollider in the same object.

Default is false.



Impacts deforming the collision shape with DeformMeshCollider option

LimitDeformationToMeshBounds

This will limit vertex deformation movement to mesh bounds.

Default is true.

RecalculateNormals

Instructs Impact Deformable to recalculate normals after each deformation.

Only deformed vertex have its normals recalculated.

Default is true.

DeformedColor and ColorDeformationDistance

Controls color deformation distance (zero (default) will disable the effect).

A deformed vertex color will be blended from the original color to the DeformedColor, by the factor of how deep deformation is (from original position [0%] to ColorDeformationDistance [100%]).

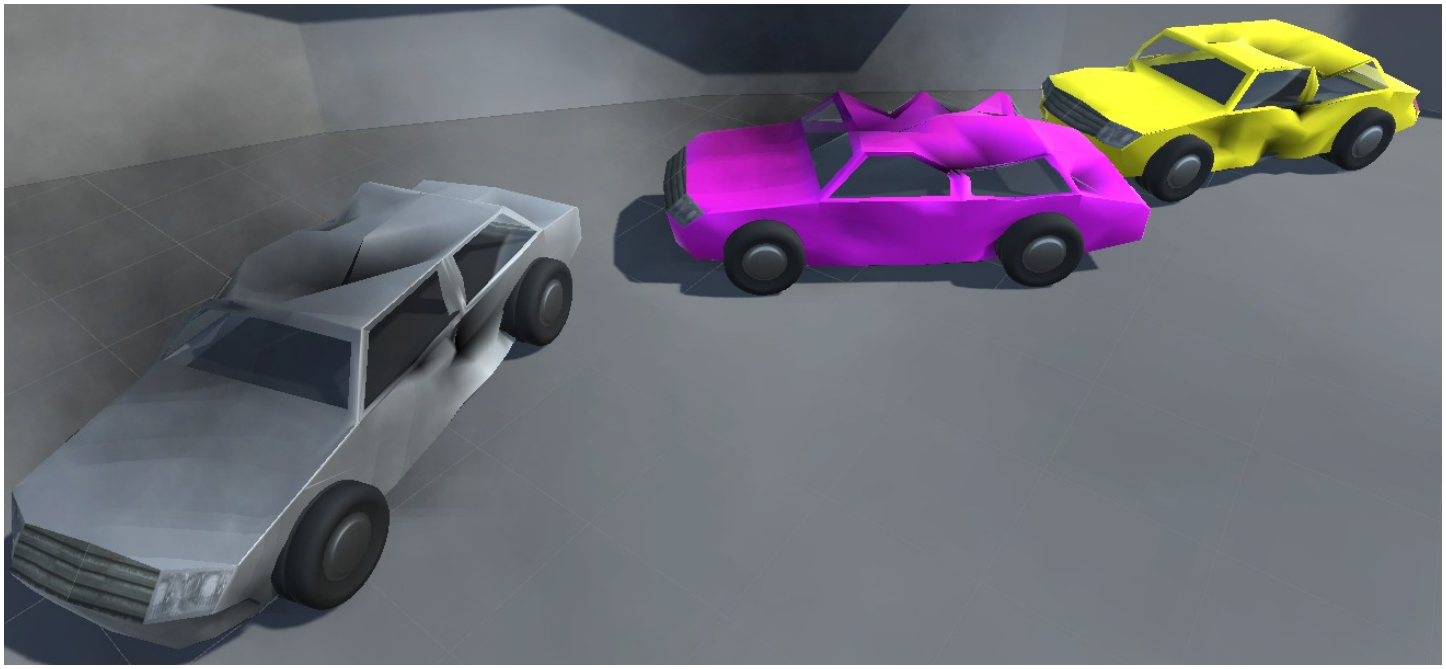
Original color is the color data from original mesh (or Color.White if none)

The original color data can also be setted with the method SetCurrentVertexColorsAsOriginal(), taking current mesh color data as base.

Please note that to be able to be tinted on deformations, the object must use a material with a shader that uses vertex color information (default Unity materials does not). In this package, there is a a simple diffuse vertex color shader for demonstration.

The repair process will revert deformed vertex color to original color.

Please check the **CarDerby.unity3d** demoscene where there is example on how tint deformation can be easily implement.



Vertex color deformation (black)

OverrideMaster

If inside a compound collider structure, all properties in child instances of Impact Deformable are inherited from the master instance (the one with a RigidBody).

This property can be used to disable this behaviour and specify different parameters (for example, specify a different hardness for each object piece).

This is used in the **CarDerby.unity3d** to set custom properties in each piece of the car.

This property is only visible if the script is in a Compound Collider structure, default is true.

Events

OnDeformForce(deformable object, point, force)

Fired at each physical deformation processed inside Impact Deformable that actually deformed something.

Please note that point and force are in object space.

Please check the **CarDerby.unity3d** demoscene where this event is used to play the "crash" sounds.

OnDeform(deformable object)

Fired when Impact Deformable applies deformation to mesh vertices.

Please check the **CarDerby.unity3d** demoscene where this event is used in bumpers script to detach from car when enough deformation is taken.

bool OnGetDeformationColor(deformable object, deformation color)

Fired when Impact Deformable will tint a deformed vertex.

The deformation color can be changed.

The result of this event controls wether to apply tint on a deformed vertex.

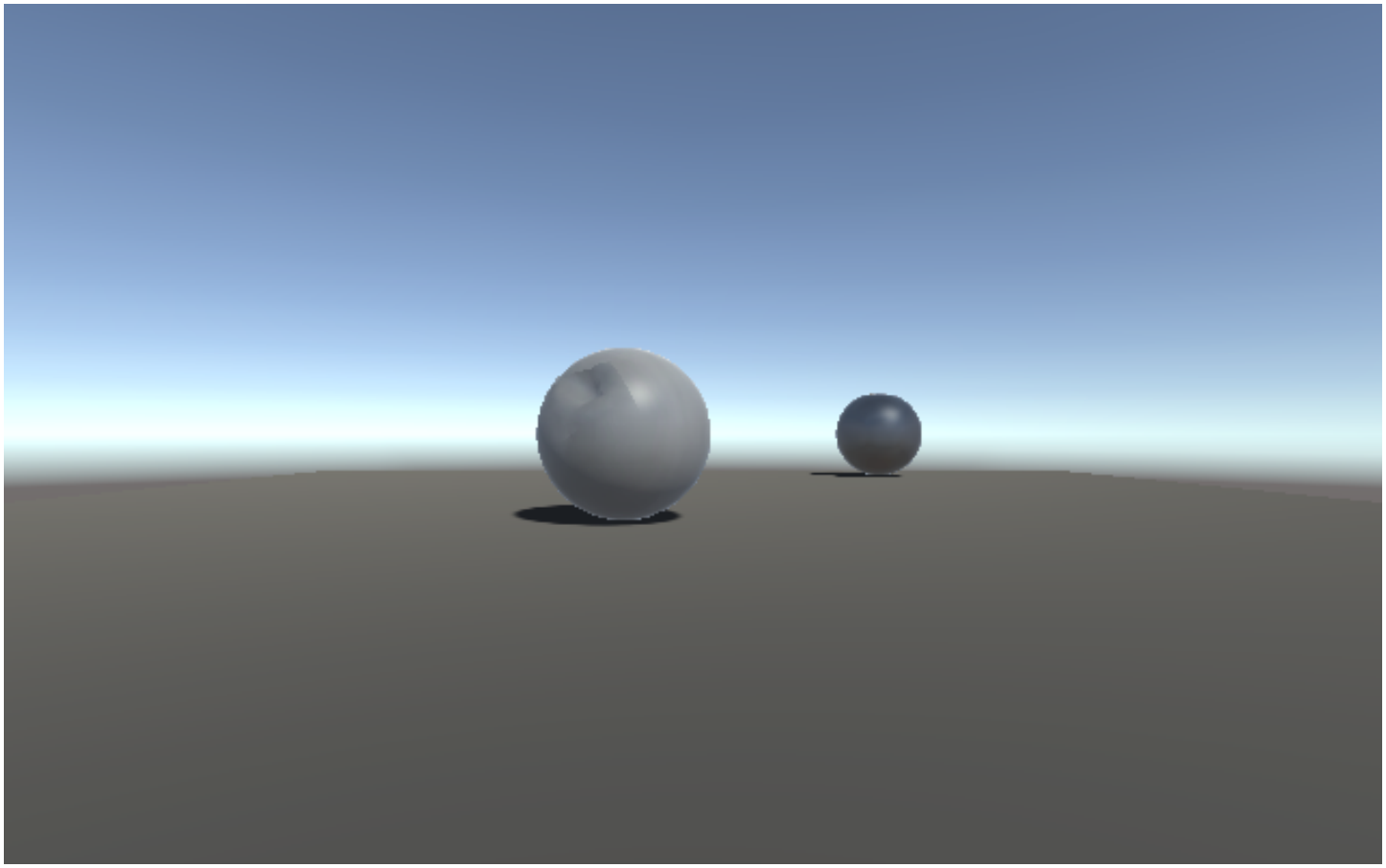
Please check the **CarDerby.unity3d** demoscene where this event is used to apply the colliding car color on deformation to mimic painting scratch.

Demonstrations Scenes

Simple.unity3d

A simple project showing the basic settings needed to have mesh deformation by impacts in one project. There is 2 standard Unity spheres with the Deformable.cs script attached, with no extra configuration.

This is a good scene to play with Impact Deformable parameters and see their effects.



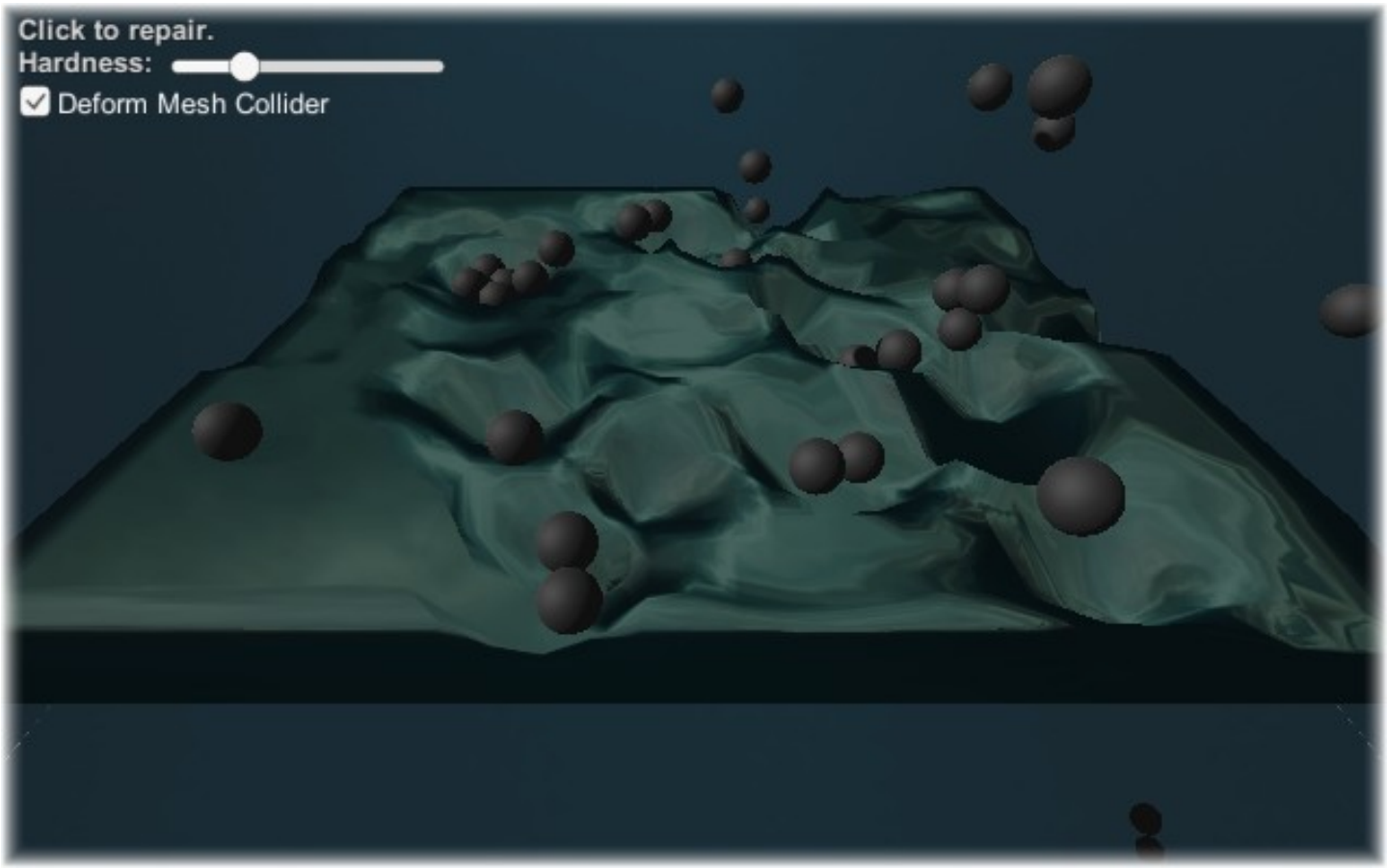
Simple.unity3d

MetalRain.unity3d

In the Metal Rain demo there is a deformable “floor” object with metal spheres falling at random positions. This demo intends to show how Impact Deformable handles many collisions per second.

There are UI controls to enable the use of Mesh Collider Deformation and configure floor hardness.

The floor object can also be repaired with mouse clicks.



MetalRain.unity3d

CarDerby.unity3d

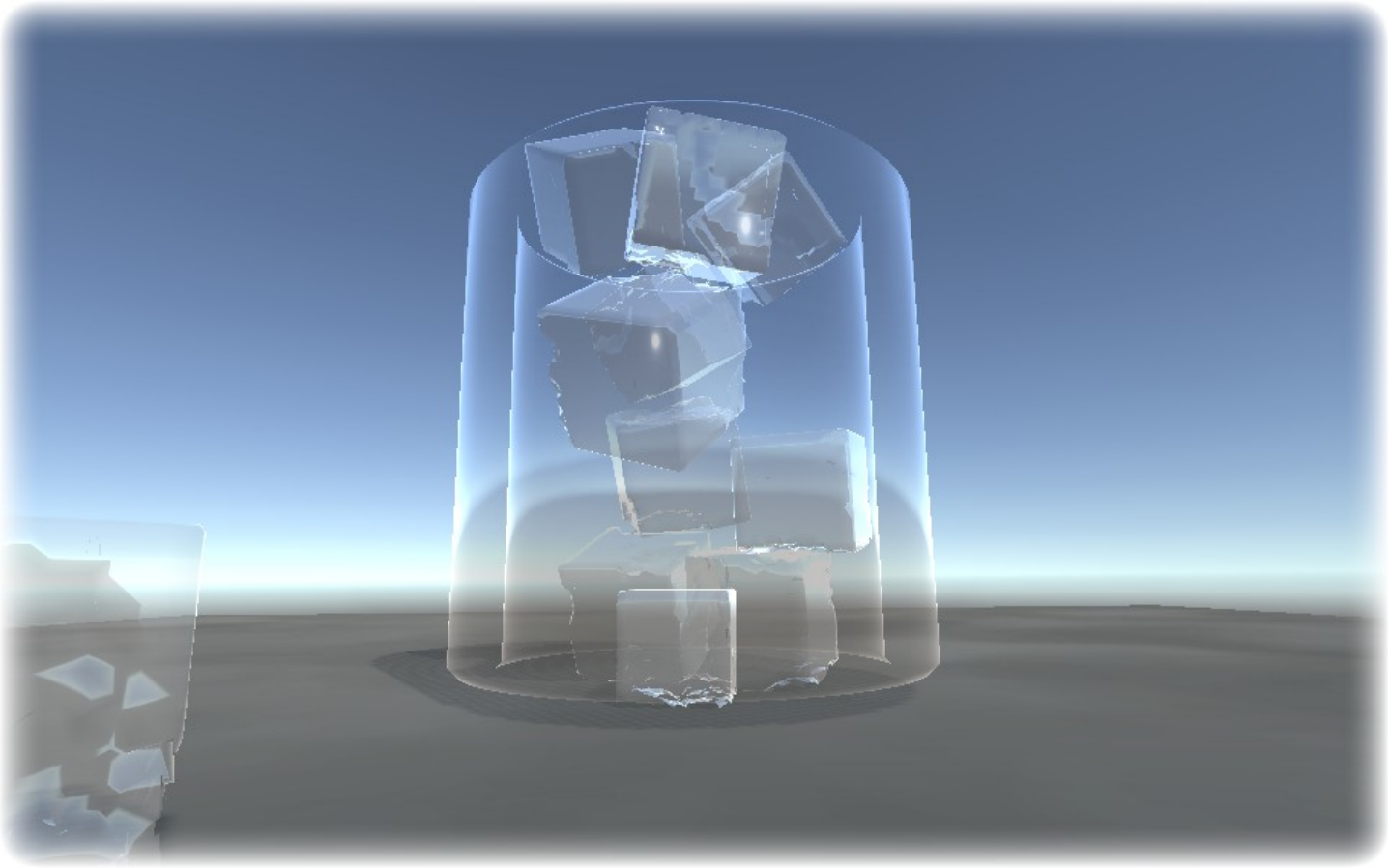
This demo is a proto game featuring a vehicle battle in a closed arena. Player can control a car with WASD/arrow keys, against 7 AI controlled cars.

Almost every aspect of Impact Deformable is demonstred somehwere in this demo (repairs, events, damage estimation, vertex color deformation, etc)

CarDerby.unity3d

IceCubes.unity3d

A simple demo with Impact Deformable applied on transparent objects



IceCube.unity3d

What is new in version 2.1

- Preciser deformations
- Full Unity 2017.3 compatibility
- Vertex color deformation
- Lot of optimizations from 6 years of user feedback
- More control over deformation with more exposed properties
- Better support for nested colliders in compound collider structures
- Damage estimation of structures based on current deformations
- Better documentation
- The old car derby demo is now an almost full game demo showing a real world use case of impact deformable.
- Ice Cubes demo

Contact

Please let me known on any problem using Impact Deformable in your projects.

Thank you and happy deforming!

L-Tyrosine
Arcadium Playware
l-tyrosine@arcadiumplayware.com