Exploder 2D Documentation

Version 1.0.0

Table of contents

1 Introduction

2 How does it work?

3 Quick start

4 How to use it in my platformer game?

6 Cracking object (pre-calculated explosion)

7 Using multiple 2D Exploders at the same time

9 Deactivation options

10 Exploder2DObject settings

13 Playmaker support

14 FAQ

15 Support

---

1 Introduction

Exploder 2D is a Unity3d plugin that allows you to explode any Sprite (Unity built-in) in **real-time**. This package is built on a successful asset Exploder: realtime mesh explosion system. The interface is similar to original Exploder package however the core has been optimized for 2D sprites.

The destruction calculation is processed in **real time**. There are no pre-calculations or predefined Sprites. Just put your beloved Sprite in the scene, Tag it with “Exploder2D” and watch the explosion!

Destroying aliens from the Demo

---

2 How does it work?

First the geometry of the sprite is extracted and passed to the mesh cutter that cuts the geometry into small pieces. For each fragment piece is created a new Sprite with same settings of the original, assigned velocity and 2d rigidbody and makes an explosion.

For best possible performance the fragments are pre-allocated in a **pool** to minimize creating and instantiating GameObjects.

Cutting algorithm is very fast however you can specify **time budget** which is the maximum time to spend on calculation in a single frame. This allows you to create a powerful explosion in few frames with **low or no FPS drop**.

---

3 Quick start

http://youtu.be/u4WCLB6quBA

The usage is **very simple**:

1. Add Exploder2D prefab to your scene
2. Adjust parameters in the inspector (or from the code)
    - radius
    - target fragments
    - …
3. Tag all objects you want to explode with the name “Exploder2D” (if you can’t see “Exploder2D” in tag list, you have to create one)
4. Move “*Exploder2D*” GameObject to desired position so it covers your objects with the red sphere radius (circle in 2D view).
5. Create a demo script with public variable to access “*Exploder2D*” and call Explode(...)
6. Watch the explosion!

For the script reference please see the attached demo examples in DemoClickExplode.cs.

This will guide you how to create a simple scene (also included in the package). Please watch the Youtube video on this address:

---

4 How to use it in my platformer game?

This package also contains an example of platformer (based on Unity demo) where character can shoot enemies and explode them.

The basic idea:

1. Add Exploder2D prefab to your scene.
2. When the enemy dies, access Exploder2D object
3. Move Exploder2D object to enemy object
4. Call Explode()
5. Watch explosion!

Example from the demo:

```
// this is called when enemy dies
void Death()
{
    dead = true;

    // access exploder singleton
    var exploder = Exploder2D.Utils.Exploder2DSingleton.Exploder2DInstance;

    if (exploder)
    {
       // move exploder on the position of this object
       exploder.gameObject.transform.position =
Exploder2D.Exploder2DUtils.GetCentroid(gameObject);
       exploder.Radius = 1.0f;
       exploder.Explode();
    }

   score.score += 1;
}
```

All **source code is included** so you can see how the demo is constructed.

---

6 Cracking object (pre-calculated explosion)

Cracking the game object is another feature that allows you to run explosion calculation = prepare the object for explosion and later immediately execute the explosion.

The **advantage** of using the *crack* is **PERFORMANCE.** You can simply run the expensive calculation a long time before the explosion and then execute the explosion instantly. This can be useful especially on mobile devices, where the performance is lower than on desktop machine and you need to object into large amount of fragments.

Usage:

Usage is similar to calling Explode(), you only need to call Crack(…) for cracking the object and later call ExplodeCracked(…). For script reference please see the attached example:

DemoClickExplode.cs

You only need to uncomment the macro ENABLE_CRACK_AND_EXPLODE on the third line to see cracking in action.

The **disadvantage** of using crack is that you cannot use more than one cracked object at the same time. You can crack only one object at the time, explode it and then crack a new one.

---

7 Using multiple 2D Exploders at the same time

Using multiple Exploder2D objects at the same time to achieve simultaneous explosions is **NOT SUPPORTED** and **NOT RECOMMENDED**.

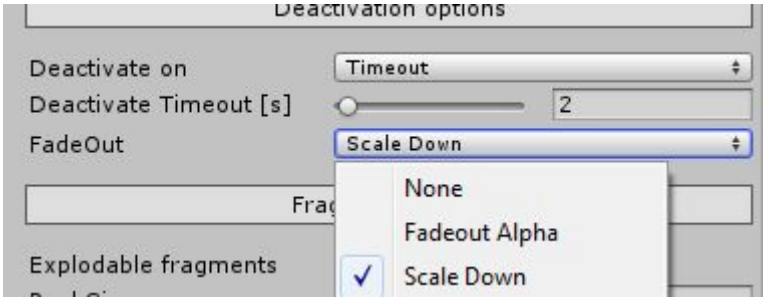
The reason why is very simple. Exploder2D calculation can be very CPU expensive, during calculation it takes as much time as specified in FrameBudget parameter. Imagine this scenario with two 2D Exploders:

- Exploder2D_1 with FrameBudget = 15 [ms]
- Exploder2D_2 with FrameBudget = 15 [ms]
- Game is running in average 30 FPS, Exploder2D_1 runs explosion and takes approximately half of the framerate, the game experience is still acceptable. Exploder2D_2 runs another explosion at the same time and takes approximately another half of the framerate. The game **FPS reaches 0 and player is experiencing lag** that can take several frames.

If you really need to run as much explosions as possible in small amount of time, you can always increase the value of FrameBudget. For example if you set FrameBudget = 100, you can easily run more than one explosion in one or two frames, however it can heavily impact your framerate.

---

9 Deactivation options



Deactivation options allows you to specify when the fragment can be deactivated.

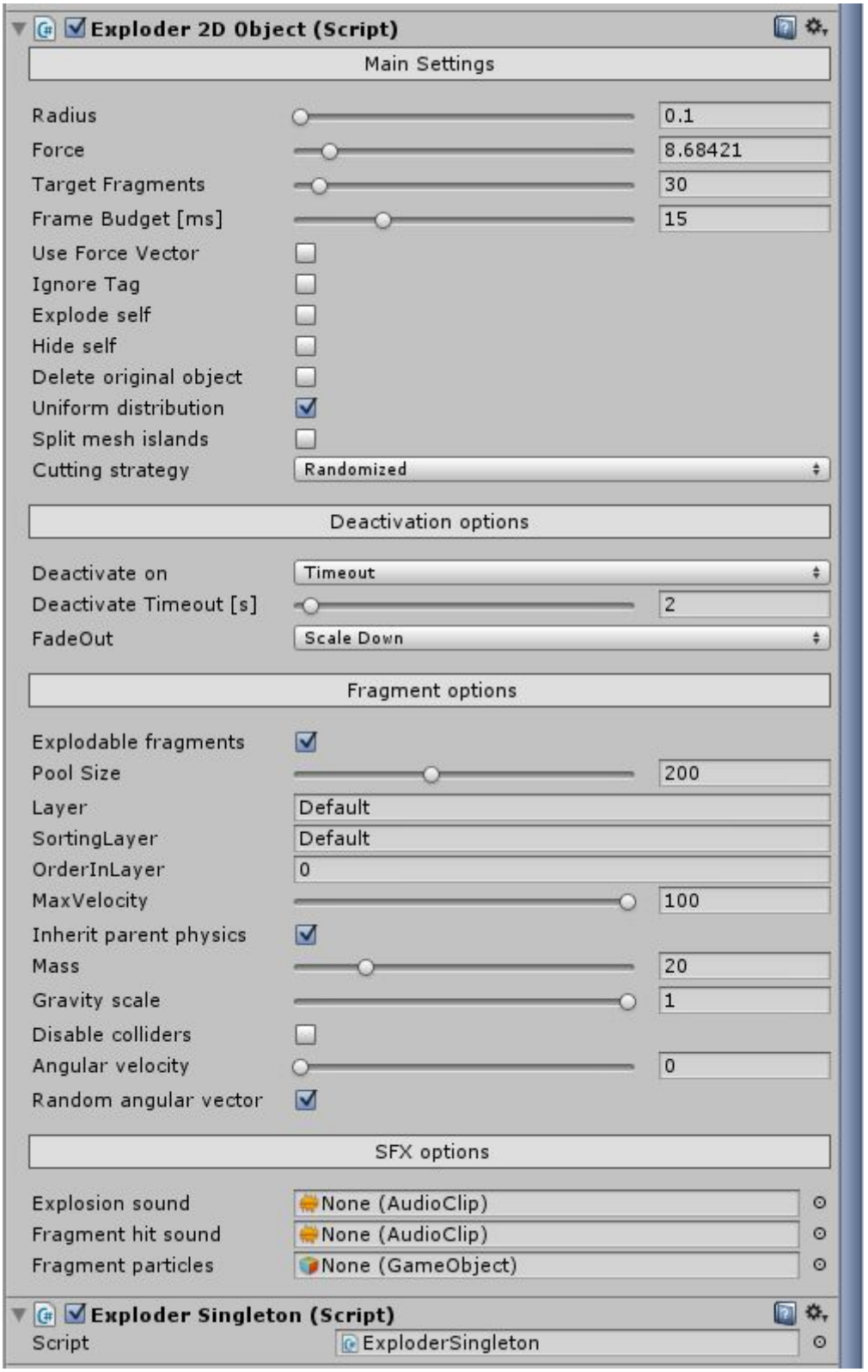
- **Never** means fragment will be active all the time
- **OutsideOfCamera** means fragment will be deactivated once the main camera doesn't see it
- **Timeout** - means fragment will be deactivated after specified time [s] after explosion

In case of **Timeout** deactivation, you can also choose whether you want to fade-out the fragments:

- **None** - no fade-out option
- **FadeOut Alpha** - fragments will smoothly change its material alpha to zero and they will become fully transparent
- **ScaleDown** - fragments will be smoothly scaled down to (0, 0, 0)

---

10 Exploder2DObject settings


Exploder 2D Object (Script)
Main Settings
Radius
0.1
Force
8.68421
Target Fragments
30
Frame Budget [ms]
15
Use Force Vector
Ignore Tag
Explode self
Hide self
Delete original object
Uniform distribution
Split mesh islands
Cutting strategy
Randomized
Deactivation options
Deactivate on
Timeout
Deactivate Timeout [s]
2
FadeOut
Scale Down
Fragment options
Explodable fragments
Pool Size
200
Layer
Default
SortingLayer
Default
OrderInLayer
0
MaxVelocity
100
Inherit parent physics
Mass
20
Gravity scale
1
Disable colliders
Angular velocity
0
Random angular vector
SFX options
Explosion sound
None (AudioClip)
Fragment hit sound
None (AudioClip)
Fragment particles
None (GameObject)
Exploder Singleton (Script)
Script
ExploderSingleton

Main Settings

Ignore Tag	Flag for not tagging Explodable objects. If you set this to TRUE you will have to assign "Explodable2D" script object to your GameObject instead of Tagging it. This is very useful if you already have tagged GameObject and you don't want to re-tag it to "Exploder2D".
Radius	Radius of explosion is a radius range in which the Exploder2D will be looking for destroyable objects.
Target Fragments	Number of fragments that will be created by cutting the exploding objects. More fragments means more calculation and more PhysX overhead and also better looking explosion.
Frame Budget	Time budget in [ms] for processing explosion calculation in one frame. If the calculation takes more time it is stopped and resumed in next frame. Recommended settings: 15 - 30 (30 frames-per-second takes approximately 33ms in one frame). However this settings depends on your game and average frames-per-second.
Force Vector (and Use Force Vector)	Note: this option is valid only if UseForceVector is true. ForceVector represents a 2D Vector direction in which the explosion fragments will be moving. For example with Vector(0, 1) exploding fragments will fly in “UP” direction.
Force	Force is how much the physical force is added to exploding fragments. More force means higher velocity.
Explode Self	Flag for destroying mesh in owner GameObject if there is any mesh component in it.
Disable radius	This option is valid only if ExplodeSelf is set to true. Disable radius scan for object, only the parent object will explode and nothing else.
Hide Self	Flag for hiding owner game object after explosion. This can be useful if you want to only hide and not to destroy owner object.
Destroy Original Object	Flag for destroying original game object after explosion. For example when you destroy an “alien”.
Split mesh islands	Option for separating not-connecting parts of the same mesh. If this option is enabled all exploding fragments are searched for not connecting parts of the same mesh and these parts are separated into new fragments.
Uniform distribution	By enabling this Exploder2D will create number of fragments per object equally regardless of object distance from the center. By default objects closer to the center (exploder2D center) will be shattered into more fragments than objects far from the

	center. Uniform distribution will guarantee that all objects will be shattered into same number of fragments.
Cutting strategy	The way how the cutting algorithm works. Currently there is only one option Randomized but more strategies will come in the next updates.

Fragment options

Explode Fragments	Flag for destroying already destroyed fragments. If this is true you can destroy the object and then all its fragment pieces. You can keep destroying fragments until they are small enough.
Fragment Pool Size	Maximum number of all available fragments. This number should be higher than Target Fragments.
Max Velocity	Maximum velocity for fragments, higher velocity will be clamped.
Random Angular Vector	This will randomize fragment rotation after explosion. If this item is false you will have specify axis of rotation manually.
Angular Velocity	Angular velocity for fragments, if “Inherit parent physics” is enabled, final angular velocity will calculated as a sum of parent and this value.
Disable colliders	Disable colliders from all fragments, use this if you don’t want the fragments collide.
Inherit parent physics	By enabling this fragment will use same physics properties as its parent rigid body. It will inherit Mass, Velocity, Angular Velocity and gravity. If there is no valid parent rigid body default settings will be used instead.
Mass	Mass of the fragment. if the parent object object has rigidbody and Inherit Parent Physics is true the mass property for fragments will be calculated based on this equation (fragmentMass = parentMass / TargetFragments)
Layer	Name of layer for fragments. Use this if you want to change default layer for fragments.
Sorting Layer	Name of the sprite layer.
Order in Layer	Order number in the sprite layer.
Gravity scale	You can scale gravity settings in range <0, 1>. Where 1 is full gravity, 0 means no gravity.

SFX options

Explosion sound	Optional AudioClip that will be played on the beginning of explosion.
Fragment hit sound	Optional AudioClip that will be played when one of the fragment piece hit another object (collider).
Hit sound timeout	This option is valid only if Fragment hit sound AudioClip is set. Timeout between AudioClips, this is useful to prevent over-halting audio system when the explosion has lots of fragments.
Fragment particles	GameObject with particle emitters, that will be run when the explosion starts. Particles will be emitted from every fragment.
Maximum emitters	This option is valid only if Fragment particles is set. Maximum number of particle emitters, use low number if your game is experiencing lag.

---

13 Playmaker support

There are 3 Exploder2D custom actions ready to work with Playmaker:

- Crack
- Explode
- ExplodeCracked

To enable these actions you have to open the action scripts located in

Assets/Exploder2D/Playmaker/*

…and uncomment the second line (//#define PLAYMAKER):

from this:

```
// uncomment next line to work with Playmaker
//#define PLAYMAKER
#if PLAYMAKER
```

to this:

```
// uncomment next line to work with Playmaker
#define PLAYMAKER
#if PLAYMAKER
```

---

14 FAQ

Q: Does Exploder 2D support 3rd party Unity plugins like 2D toolkit?

A: No, it doesn't. Exploder 2D is exclusively made only for Unity built-in Sprites. The reason is that the other plugins like 2D toolkit are using standard 3d mesh (plane). For this case you can use the original Exploder package as it works with 3d meshes.

Q: I cannot explode my objects. Sometimes it creates only one big fragment.

A: Make sure your objects are in exploder2D radius, properly tagged and the TargetFragments value is high enough. If it still does not work, please contact me by e-mail with screenshot of your object hierarchy and your model in scene.

Q: Fragments are shaking after explosion.

A: This is a behaviour of the 2D collision system in Unity. You can adjust it by going into Edit -> ProjectSettings -> Physics2D. There are many variables you can play with, but to fix the shaking you can try to decrease the Baumgarte Scale and Baumgarte Time Of Impact Scale values.

Q: Does Exploder2D support reverse explosion?

A: Unfortunately not, Exploder2D can only explode objects in usual way.

Q: I don't want to change the tag to "Exploder2D", it already has one.

A: You don't have to, just make sure to enable "Ignore Tag" and add "Explodable2D" component to your game object.

Q: Can Exploder2D destroy only a portion of the object, like a head or arm of a character?

A: Unfortunately it can't, Exploder2D works only with a whole sprite, in case of a character it will explode head, arm, body, legs and everything else at once.

Q: How to access fragment pool and get active fragments?

A: Please refer to file ***HowToGetActiveFragments.cs.***

Q: Can I use Exploder2D to create fragment prefabs?

A: No, Exploder2D is only for real-time use. However you can use Crack() feature to pre-calculate explosion and execute it later.

Q: Can Exploder2D be used with Unity 4.x?

A: No, Exploder2D requires to access Sprite mesh which is only Unity 5.x+ feature.

Q: Any tips for increasing performance?

A: First of all **TargetFragments,** try to keep it low, less fragments mean less overhead and less calculations. Also make sure that **FrameBudget** is lower than half of your expected frame rate, for example game with 30 FPS should have **FrameBudget** 15 and less, otherwise you can notice lagging. Also don't use **SplitMeshIslands** so much, it can save you few more frames. Don't forget that performance is always about tradeoff, good luck with your game!

15 Support

If you have any questions or need a help with setting up the Exploder2D game object you can write to unity forum:

http://forum.unity3d.com/threads/released-exploder-2d-realtime-sprite-explosion-system.365647/

or you can send me an e-mail to:

gamesreindeer@gmail.com