EXPLODER

Reindeer Games

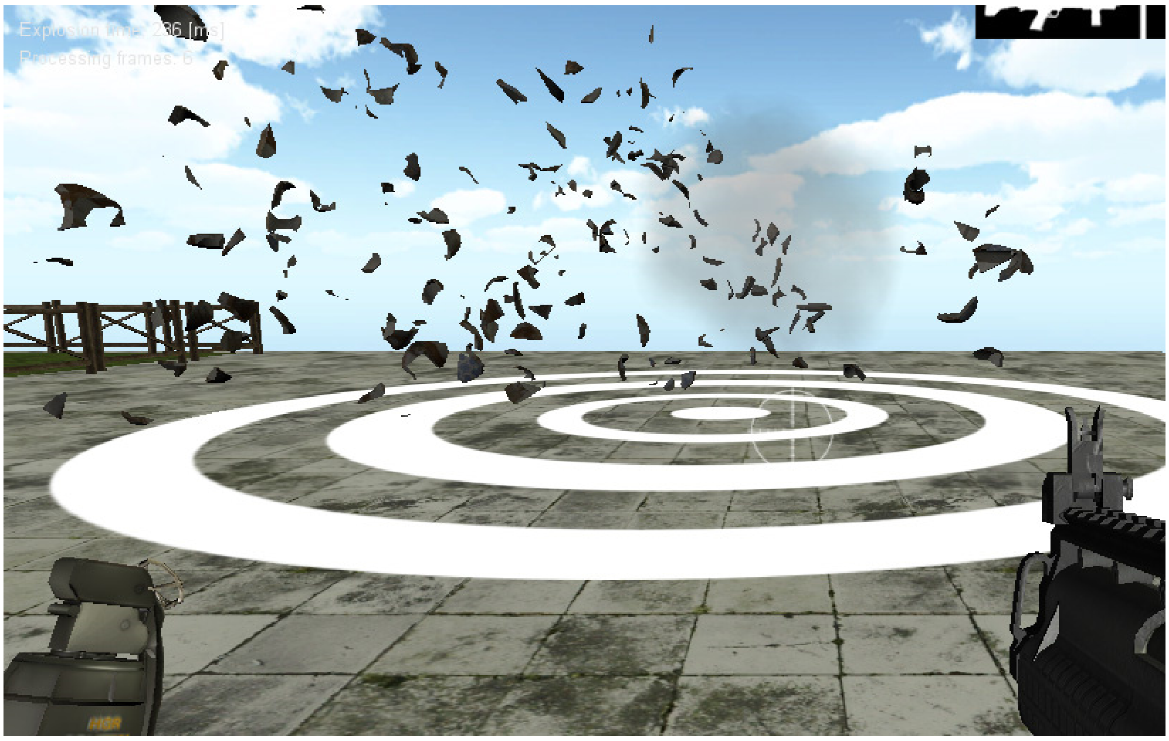
Version 1.2

Thank you for purchasing **Exploder!**

About

Exploder is a Unity3d asset library that allows you to destroy any GameObject that contains a mesh in **real-time**.

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



```
Explosion time: 286 [ms]
Processing frames: 6
```

Destroying vases from the Demo

How does it work?

The library contains a powerful **mesh cutter** that finds a mesh in your GameObject. After that it cuts the mesh recursively to small pieces, assigns a rigid body and velocity to each of them 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**.

The usage is **very simple**:

1. Find a script component **ExploderObject.cs**
2. Assign it to any GameObject (empty or not-empty)
3. Adjust parameters in the inspector or from the code (radius, number of fragments, …)
4. **Tag** all destroyable objects with “Exploder”.
5. Move the GameObject with **ExploderObject** component to desired position.
6. Call function Explode()
7. Watch the explosion!

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

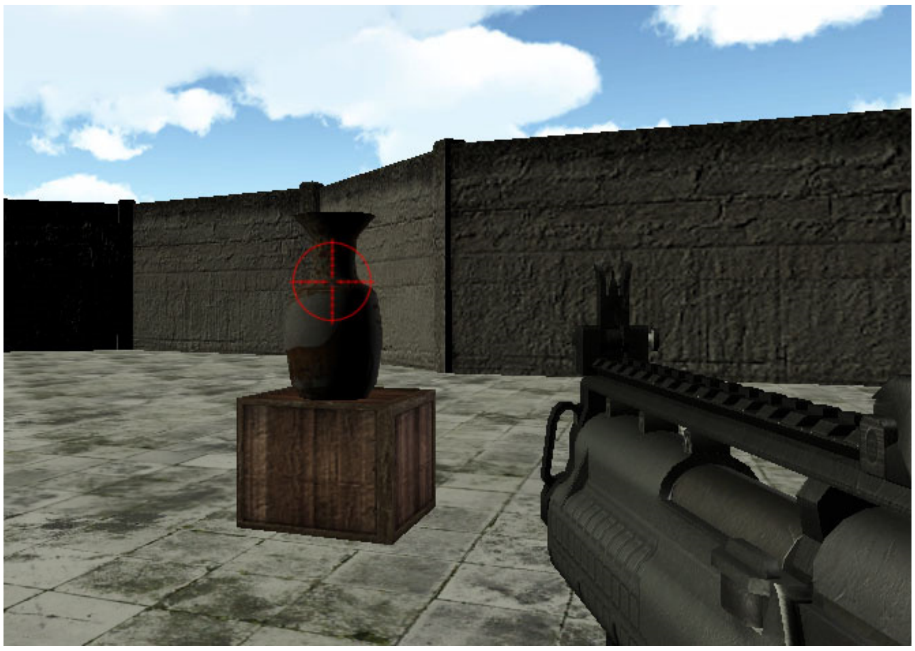
How to setup a simple scene?

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

http://youtu.be/u4WCLB6quBA

How to use it in my FPS game?

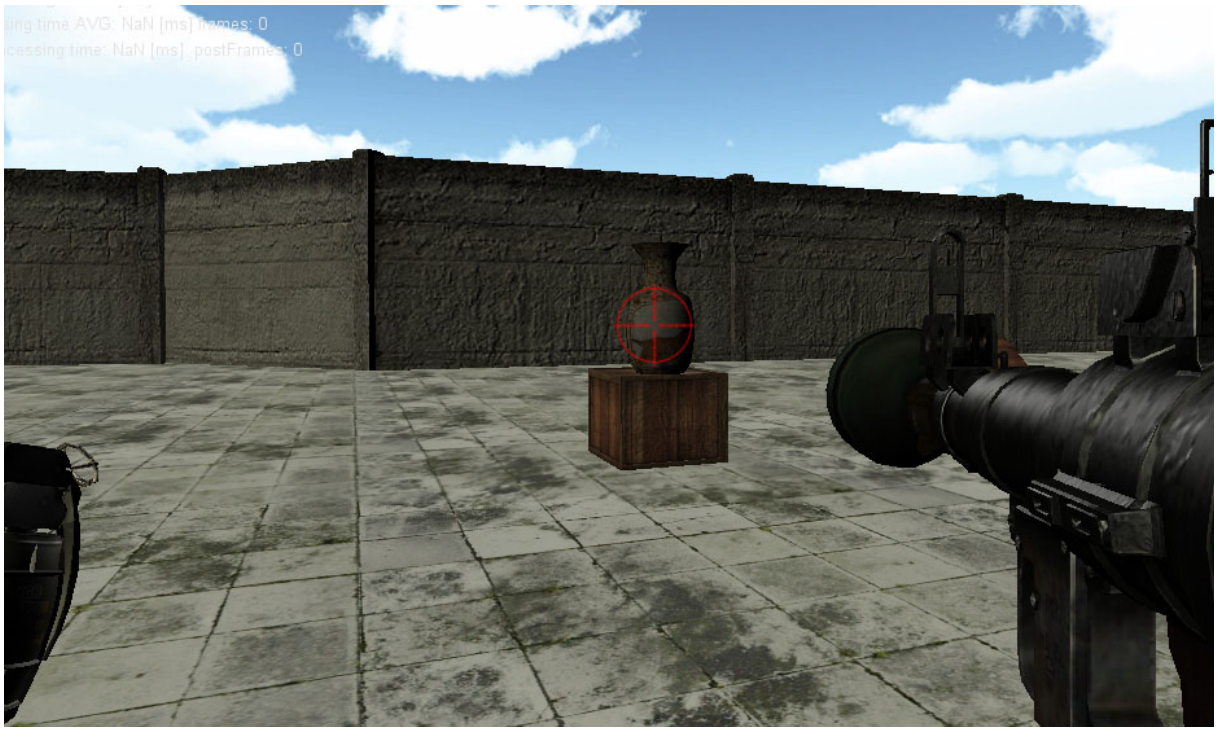
This package also contains a first person shooter demo to demonstrate how the Exploder can be used with various weapons.



Shotgun example from the Demo

The basic idea:

1. Create empty GameObject with ExploderObject component.
2. (Setup your weapon and camera) and shoot with your mouse.
3. At the same moment run a raycast in direction from the camera.
4. Get an intersecting GameObject (ex. Vase).
5. Move ExploderObject to the position of the intersecting GameObject (ex. now the ExploderObject has the same position as the Vase).
6. Call Explode()
7. Watch explosion!



```
sing time AVG: NaN [ms] frames: 0
cessing time: NaN [ms], postFrames: 0
```

RPG example from the Demo

The setup for the RPG or Grenade is very similar. Just shoot a rocket (or throw a grenade) and when it hits the object (or after a timeout) move your ExploderObject to the same position and call Explode().

You don't need to maintain several ExploderObject components. Just only one and always move it to desired position.

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

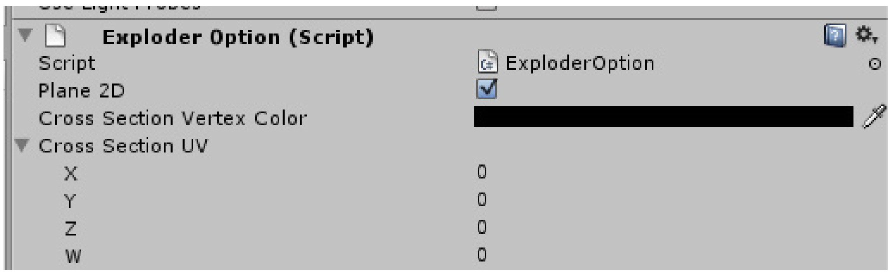
Exploding 2D sprite (Plane mesh)

Tested with 2D Toolkit.

(For the 2D physics support, see the 2D Physics chapter)

Exploder also works very well with 2D sprites or plane meshes. The only thing you have to do is to mark the object in **ExploderOption** settings.

ExploderOption is another script object you can find in the asset folder together with ExploderObject. You have to **assign this script** to your game object (sprite) you want to explode. There are several options in the script:



Exploder Option (Script)

Script	ExploderOption
Plane 2D	✓
Cross Section Vertex Color	■
Cross Section UV	
X	0
Y	0
Z	0
W	0

As you can see there is a checkbox *"Plane 2D"*. By selecting it you specify that this object (owner game object) is a 2d plane or a 2d sprite. Later when you run the explosion Exploder will look for the ExploderOption and if it is marked as 2d plane it will change the explosion calculation to work with 2d meshes.

In short:

1. Find ExploderOption script and assign it to your 2d game object
2. Check the *"Plane 2D"* option
3. Run the explosion!

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.

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.

Using multiple Exploders at the same time

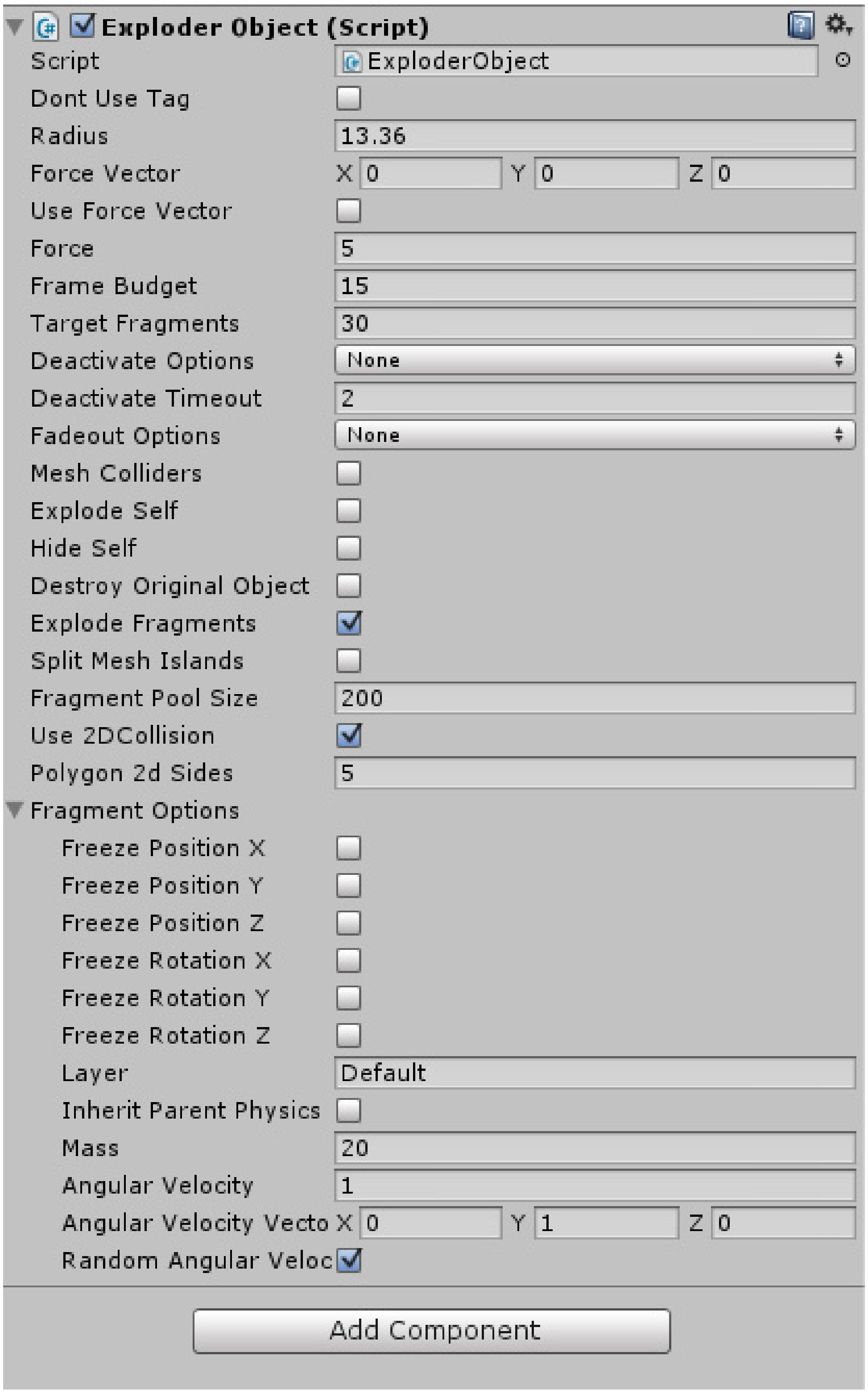
Did you ever thought about using multiple Exploder objects at the same time to achieve simultaneous explosions? Some of you already did however this feature is **NOT SUPPORTED** and **NOT RECOMMENDED**.

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

- Exploder_1 with FrameBudget = 15 [ms]
- Exploder_2 with FrameBudget = 15 [ms]
- Game is running in average 30 FPS, Exploder_1 runs explosion and takes approximately half of the framerate, the game experience is still acceptable. Exploder_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 minimum 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.

Settings for ExploderObject



Exploder Object (Script) ✓

Script	ExploderObject
Dont Use Tag	☐
Radius	13.36
Force Vector	X 0 Y 0 Z 0
Use Force Vector	☐
Force	5
Frame Budget	15
Target Fragments	30
Deactivate Options	None
Deactivate Timeout	2
Fadeout Options	None
Mesh Colliders	☐
Explode Self	☐
Hide Self	☐
Destroy Original Object	☐
Explode Fragments	✓
Split Mesh Islands	☐
Fragment Pool Size	200
Use 2DCollision	✓
Polygon 2d Sides	5
Fragment Options	
Freeze Position X	☐
Freeze Position Y	☐
Freeze Position Z	☐
Freeze Rotation X	☐
Freeze Rotation Y	☐
Freeze Rotation Z	☐
Layer	Default
Inherit Parent Physics	☐
Mass	20
Angular Velocity	1
Angular Velocity Vecto	X 0 Y 1 Z 0
Random Angular Veloc	✓

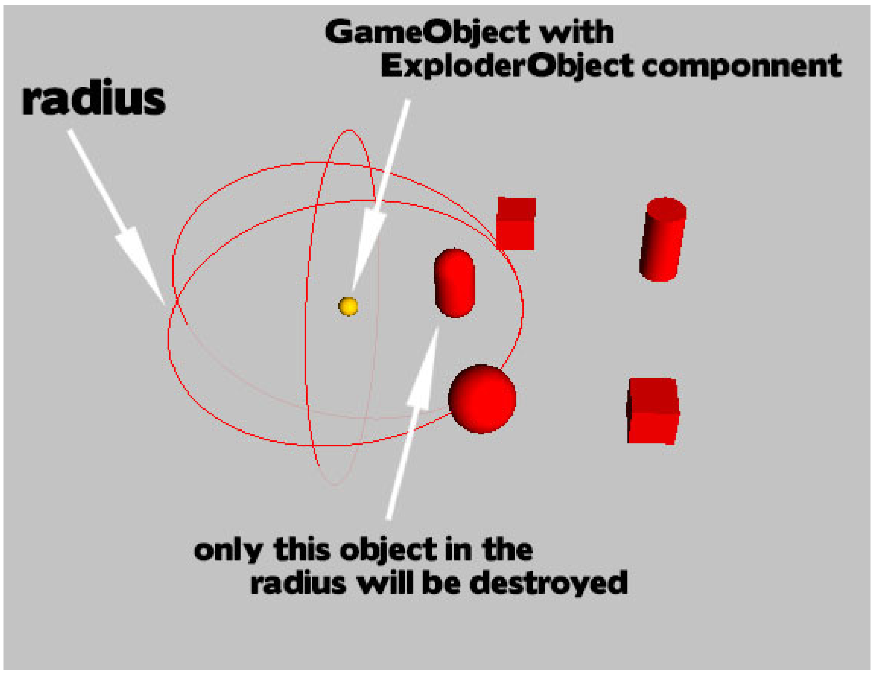
Exploder object from inspector view

DontUseTag

- Flag for not tagging Explodable objects. If you set this to TRUE you will have to assign "Explodable" 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 "Exploder".

Radius

- Radius of explosion is a radius range in which the Exploder will be looking for destroyable objects. Notice the Red wire-frame gizmo in scene view.



Radius in scene view

ForceVector (and UseForceVector)

- Note: this option is valid only if UseForceVector is true. ForceVector represents a 3d Vector direction in which the explosion fragments will be moving. For example with Vector(0, 0, 1) exploding fragments will fly in "UP" direction. In the Demo Shotgun weapon is using ForceVector in direction of the camera, so the fragments are always moving from the player.

Force

- Force is how much the physical force is added to exploding fragments. More force means higher velocity.

FrameBudget

- 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.

TargetFragments

- 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.

VisibilityCheck

- If this flag is true all fragments outside of camera frustum are deactivated. This is useful for better performance.

MeshColliders

- Using mesh colliders for fragments. Use this option wisely because mesh colliders are very slow and you should avoid them in most cases. Mesh colliders can be very useful if you want to run explosion with small or zero force.

ExplodeSelf

- Flag for destroying mesh in owner GameObject if there is any mesh component in it.

HideSelf

- Flag for hiding owner game object after explosion. This can be useful if you want to only hide and not to destroy owner object.

DestroyOriginalObject

- Flag for destroying original game object after explosion. For example when you destroy a “chair” model, it will cut the chair to small pieces and create new fragment game objects. If this flag is true it will call Object.Destroy(chair) and destroy the original chair GameObject from the game. Otherwise the original object (chair) will be only deactivated.

ExplodeFragments

- 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.

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.

FragmentPoolSize

- Maximum number of all available fragments. This number should be higher than TargetFragments.

2D Physics (Unity4.3+)

Tested with 2D Toolkit.

Exploder support 2D physics as well. You can enable this feature by setting Use2Dcollision to True. All explosion fragments will have 2D colliders (PolygonCollider2D) and it is important that you will not use this setting with exploding 3D objects (it might work but it might lead to unexpected behavior).

http://youtu.be/IZWz5mt2kIY

IMPORTANT NOTE:

Exploder DOES NOT work with Unity4.3+ built-in sprites, because there is no way to access sprite mesh data required by Exploder.

Playmaker support

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

- Crack
- Explode
- ExplodeCracked

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

Assets/Exploder/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
```

Support

If you have any questions or need a help with setting up the Exploder GameObject you can write to:

http://forum.unity3d.com/threads/190198-Exploder-RELEASED

or you can write me an e-mail to:

gamesreindeer@gmail.com