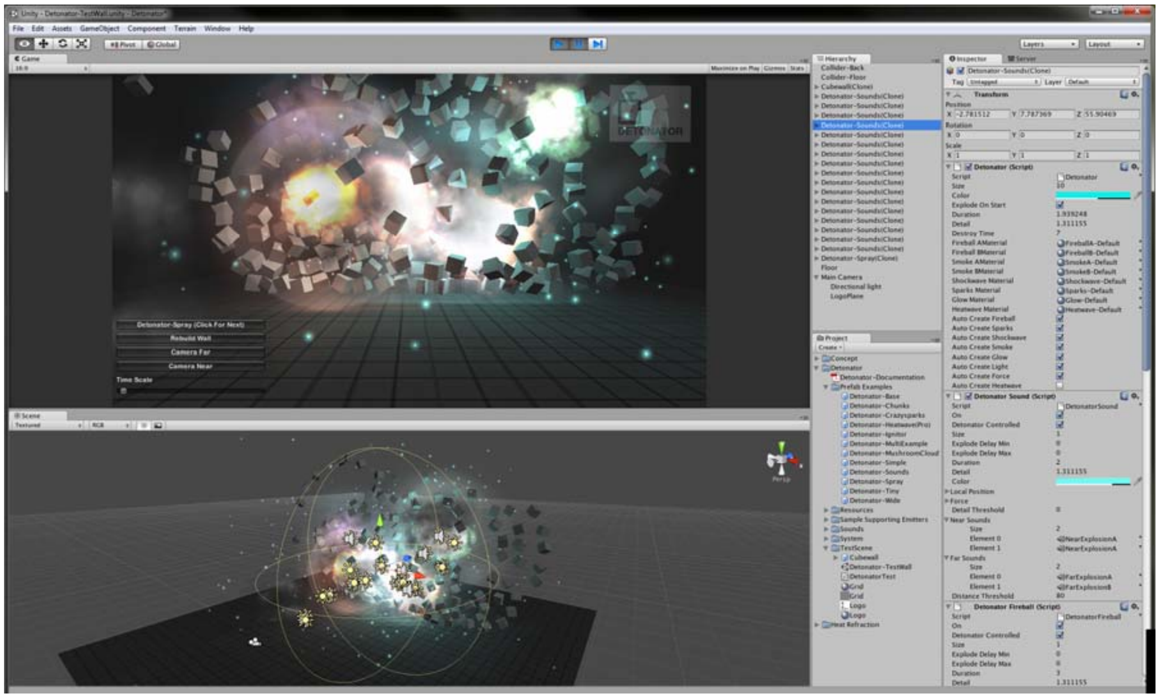
Detonator

Parametric Explosions for Unity

Detonator was created originally for the Unity Summer of Code 2009 by Ben Throop (http://variancetheory.com) over the course of 6 weeks, with initial release in early September 2009. It has been tested on Unity Indie and Pro for Mac, PC, and Webplayer deployment. It is currently untested on iPhone, Wii, or any other platforms.

Detonator lets you make good looking explosions quickly and easily. How you use it depends on who you are and what your goals are. Solo coders can quickly get prototype explosions going while artists can stack effects to quickly make complex explosions. In this short document, we'll take a broad view of what's included in the package first, and then look at different ways of getting started.



So first, here are the different folders included in the package:

Prefab Examples

These examples are configured GameObjects with Detonator and one or more Detonator Components applied.

1. **Detonator-Base** – A straight out of the box Detonator. Same as you'd get by just attaching a Detonator component to any GameObject and letting it blow up.

2. **Detonator-Chunks** – Shows the DetonatorSpray component emitting some GameObjects that include smoke trails.
3. **Detonator-Crazysparks** – Shows a configured DetonatorSparks object that shoots out a ton of sparks in a flattened pattern.
4. **Detonator-Heatwave(Pro)** – Demonstrates the heat-distortion effect. Only works in Unity Pro, not Indie.
5. **Detonator-Ignitor** – This effect shows a configured DetonatorForce which in addition to causing a RigidBody explosion also ignites any Rigidbodies it hits.
6. **Detonator-Insanity** – A demo effect that puts everything into a single package.
7. **Detonator-MultiExample** – Shows a single Detonator with three separate DetonatorFireballs, all different colored, positions, and sizes.
8. **Detonator-MushroomCloud** – Shows an example of a complex effect, a nuclear blast.
9. **Detonator-Simple** – Shows what happens when you replace all of the Detonator materials with a single glowing dot and turn off more intensive components.
10. **Detonator-Sounds** – Demonstrates the DetonatorSound component, which lets you have lists of sounds that are randomly chosen to play, and also are different based on whether the explosion is near or far. If you use the Test scene described below, you can switch your camera distance between near and far easily to see the difference between the near and far sound.
11. **Detonator-Spray** – Similar to Detonator-Chunks, but emits particle emitters which have RigidBody components.
12. **Detonator-Tiny** – Demonstrates the effect Detonator's size parameter.
13. **Detonator-Wide** – 3 Fireballs arranged horizontally for a wider effect.

Resources

These are the default textures that Detonator uses when building its materials on the fly. You can replace these or just make new materials pointing to other textures. Since these are in /Resources they'll automatically get included in any webplayer build so watch out for that!

Sounds

Explosions aren't very effective if they are silent, so some sample sounds are included. These sounds are in the public domain and were found in a sound pack at: http://www.freesound.org/packsViewSingle.php?id=4366

Sample Supporting Emitters

These emitters are used by other effects, like DetonatorSpray and DetonatorForce to do various neat things.

System

This is the actual Detonator code. You can drag components directly from here or use the Component menu.

TestScene

This folder includes a scene that lets you easily test your explosions. Open up Detonator-TestWall to try it out.

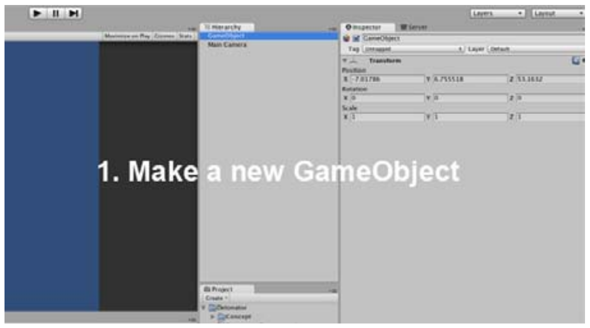
Getting Started

So, that’s what’s in the package. How should you start out? Kind of depends on what you want…

A) Show me something cool now! (using prefabs)

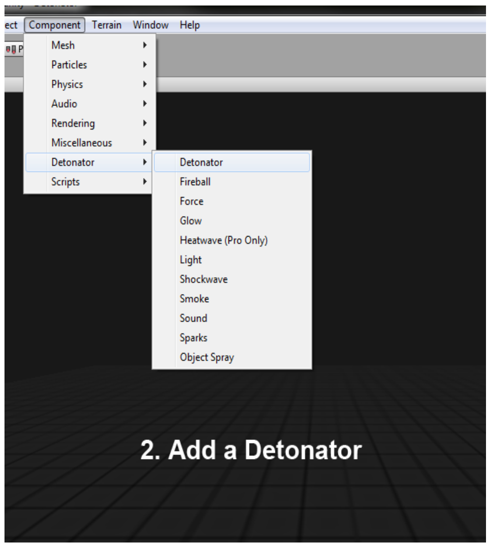
1. Make a new project and then import the Detonator package if you haven’t already.
2. Then open up the TestScene folder and double click the scene Detonator-TestWall.
3. Hit Play.
4. Click in the scene.
5. Hit the buttons.
6. You get the idea.
7. If you feel adventurous, try altering the parameters on the prefab you’re currently exploding to see what happens.

B) How does this work? (making your own)

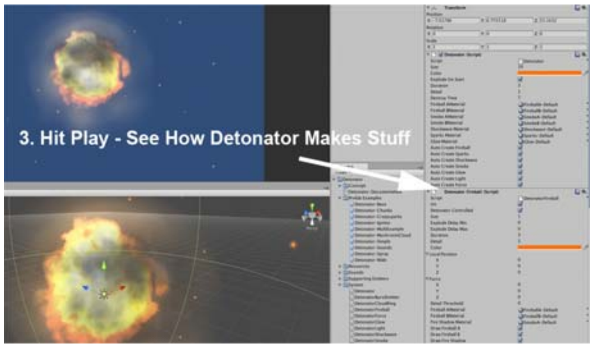


1. Create an empty GameObject in your scene or choose an existing one. If you are unsure of how to do that, see option A. 😉
2. From the Component Menu, choose Detonator->Detonator. (*if this doesn’t show, try restarting Unity…)*
3. Hit Play.
4. Change up the parameters. Notice that changing size makes the entire explosion larger or smaller? This is one of the main benefits of the system… this is normally hard to do by hand.

5. Try changing the color. Notice that the color of all the pieces are affected? The alpha of the main color is the influence over the subcomponents. Another main design goal.



6. So, almost all of the pieces of the explosion get created on the fly with no intervention needed. However, if you want to customize a piece, you can add a DetonatorFireball or DetonatorSparks to the same GameObject. For instance, try choosing Detonator->Sparks from the Component menu. If you don't change it at all, it will look the same, because its using the default settings. But now try turning Detail up to 10. That means that 10x as many particles will spawn as the default emitter.
7. So now you have a crazy 10x spark emitter. Well, the main Detonator is the boss of all the sub components. So now turn the Detonator's detail down to .5. That will in turn affect the Sparks' detail, cutting down its particle count. Once you have a lot of pieces going, its very convenient to be able to tune the detail of the entire effect by changing the main Detonator's detail level.



8. Try mixing and matching components and parameters and see what happens. Do note that some components have inherited parameters that don't make sense… for instance, the Color parameter on DetonatorSound. Eventually, this will be cleaned up but for now, just know that not every parameter does something.
9. CAUTION: If you add DetonatorComponents to a prefab directly in the Project panel, they will not receive their default values (Reset() does not run). This can lead to effects not working properly. So **while building effects, you should always add components in the scene first, and then save to a prefab**. If there's a workaround for this it'll be added later on, but you should know that now.
10. Once you've made your own, save it as a prefab. If you'd like to see what it looks like in the Test Scene, select the Main Camera in that scene and drag your prefab onto the Detonator Test script. You can replace the Current Detonator, and/or one of the slots in the Detonator Prefabs list.

C) Can I work with Detonator via code?

Sure! Some users will be more comfortable creating prefabs for their Detonator explosions, but you can just as easily do things in code. This can be handy when you, say, want to tie the color or size of your explosion to the amount of damage dealt, or the area hit on a vehicle. Basically, add the Detonator component to your GameObject. Then set parameters… Then it's up to you, you can have the Detonator explode on start if the explodeOnStart() parameter is true (which it is by default). Or you can hold off and call Explode() when you want. However, if you do this then you want to set the destroyTime parameter to zero so it will live forever… otherwise your gameobject will destroy itself after n seconds. Most of the time this is convenient, but you should know it's happening when your objects randomly disappear.

To reiterate, there's two usage cases for code. One is to let explodeOnStart = true cause Explode() to get called automatically. The other way is to set that to false and call Explode() yourself. The first method is probably the more common one, but there may be a use for the second one. We'll see.

D) Can I use my own materials and textures?

Yes. While a Detonator will build its own default materials, you can easily use your own by just making the Detonator and dragging your materials into the corresponding slots. The 8 slots all cascade down to subcomponents, so you don't need to create them if all you want to do is change materials. The Detonator-Crazysparks prefab demonstrates these material overrides. However, animated textures are not currently supported. It wouldn't be terribly hard, but it's not in the first release.

What's going on under the hood (the very short version)

Detonator is the main component that is the "master" of the explosion. It can generate, and/or control DetonatorComponents. DetonatorComponents all inherit common properties, like size, color, and duration. Detonator is responsible for altering these as needed, so the user can say the Detonator size to 5m and all pieces will follow accordingly.

DetonatorBurstEmitter is the one exception. It is a wrapper for standard Unity particle components and includes a lot of common functionality in each of the other components. So DetonatorFireball for instance just creates and controls 3 DetonatorBurstEmitters with different properties and materials. So in that regard it's a 4 layered system (at the component level):

Detonator
DetonatorFireball
DetonatorBurstEmitter x 3
Particle Emitter, Animator, Renderer x 3

Conclusion (for now)

There's a lot more to talk about but it's time to put the system in your hands and see what happens. If you find bugs or need new features I'm committed to making the system a valuable one in the Unity Community. You can contact me at ben@emergentbehavior.com or Twitter @benimaru.

PS, I'd like to thank my wife Kristin and my new son Elliot for their love, patience, and support during these intense six weeks.

I love you both!