Figure 1

By Paul Fitzpatrick

## Introduction

I enjoy using Blender to make fun little pictures and animations for birthdays or holidays. I imagine I'm not the only one. One day I thought: wouldn't it be neat if you could put Blender designs online in an easy-to-use form, so that anyone (not just Blender users) could customize them for their own use? This idea slowly became a website called MakeSweet, and I'd like to talk about how it evolved and what I learned along the way.

***Figure 1:*** *A selection of designs from MakeSweet. The designs take an image supplied by the user (here I used the Blender logo) and produce a picture or animation based on it. There are some soft body animations, lens effects, and lots of photo backdrops with 3D surfaces overlaid. 90% of the designs shown here are user-created (the rest I made).*

### The world of online generators

Online tools that help you make things are called generators. For example, there are generators that help you make buttons, write love letters, construct plausible excuses, and so on. Within the world of generators, visual effects are popular. Who wouldn't want to put their photo on currency, or see it on a billboard?

When I looked around, I found there were already plenty of sites out there for creating visual effects. But I guessed that the openness of Blender might let me go further than other similar sites. Adapting a free and open source tool to new uses is much more pleasant than working with closed products, because there are fewer roadblocks to integration. It is also a wonderful base to build a community around, since it can be freely shared. So I hoped that, if I did it right, I could enable a community of generator-makers, rather than just create generators myself. And in fact that has started to come together (see examples in Figure 1), though in a form I didn't anticipate. I'll explain, but first let me talk about the basic technical and artistic challenges of using Blender to make generators.

### The technical challenge

Blender is a wonderfully adaptable program used by everyone from movie-makers to architects. To adapt it for online use, the main problem is time. It takes time to render pictures and animations, but on the internet there is a strong expectation of immediate response. Let's think about what properties a generator site needs to have:

1. Fast to respond. Something cool must happen within seconds of a user's request.
2. Cheap to run. The site must not cost too much to operate.
3. Scalable. The site must remain fast to respond and cheap to run as its popularity grows. Otherwise the only possible fates for the site would be to either self-destruct or remain forever obscure, neither of which is particularly desirable.

To satisfy these requirements with Blender, we're going to need to be ruthless about baking, caching, and otherwise pre-computing everything we possibly can. Ideally, we would also offload as much computation onto users' own computers as possible

by Paul Fitzpatrick

There is a Blender browser plug-in that can be made to work, but expecting users to install software in order to use a generator site is a bit too much to ask. The most practical solution is to do all the Blender work on a server (or set of servers) and send the results as images or animations to a user's browser. So we really need to reduce the amount of work done for each user request to the bare minimum.

How far we can go with pre-computing depends on what kinds of user customizations we want to support. I decided to restrict my attention to a class of designs and customizations that was wide enough to allow interesting generators, but restricted enough to allow almost everything to be pre-computed. The customizations I decided to support were the replacement of image textures with a new image of the user's choosing. This is a good match for user expectations ("upload a photo and see something fun happen"). The designs I decided to support were those in which the image textures under user control do not affect geometry and do not overlap. This rules out some possibilities, for example 3D text or displacement maps, but still leaves a range of useful effects, as we'll see.

***Figure 2:*** *An animated flag design with one image texture under user control (the surface of the flag). Prior to online use, the coordinates that a test image projects to in a rendered result are probed and recorded, for each frame in the animation. This allows fast stills and animations to be generated when a user supplies a photo.*

To get technical, the requirement I chose was that the appearance at each location in a frame of the output render should be a function of the appearance of at most one location in the input image (I'll also assume the function is linear or can be approximated as linear, although this isn't as important). Consider the flag design in Figure 2. Here, the sky is fixed – it is not affected by the input image.

The appearance at each location of the render within the flag's boundaries is drawn from a single location in the input image, although exactly which changes from frame to frame as the flag billows. Under these conditions, it is easy to get Blender to pre-compute how the output render is affected by the input image, and to store this mapping efficiently.

This can be done at leisure, offline. Online, then, for each user request, I use a stripped-down optimized renderer I wrote called the "Mixer" (meant to sound like a cheap generic knock-off of "Blender") to apply this mapping to the user-supplied image without needing to run Blender. The quality is not quite as good as Blender would give due to sampling issues, but with it, users then have a good sense of whether the generator is giving them something they want, and can go on to make a high-resolution version or an animation. For a high-resolution result, I expect that users will be sufficiently motivated to be willing to wait for a few more seconds, so I run Blender on the server for them (at a low priority so as not to hurt response time for other users).

by Paul Fitzpatrick

For animations, I generate an animated GIF produced by concatenating preview results; using Blender for an animation would take a very long time, and the extra quality of renders would not be worth much once jammed into a 256-color GIF palette (also, individual frame quality matters less in a moving sequence).

There are lots of other possible choices one could make to get an efficient family of generators. This is just the one I chose to start with, and it has worked out pretty well in practice. The key point is to support a wide enough range of effects to be interesting, but be constrained enough to allow fast rendering (see Figure 3).

***Figure 3:*** *For speed, MakeSweet restricts designs to cases where each location in the output is a function of at most one location in an input image. There's plenty of scope for creativity within the limits of what the website can quickly render. It is fine for the user's image to appear multiple times, either by having the same material on many objects (see top left, from a St. Patrick's day pot of gold animation) or by reflection (see top right, from a New Year's day animation). Distortions are also fine (see bottom left, from a Valentine's animation, where the user image is seen through a heart-shaped lens). And transparency can work too (see bottom right, a pumpkin carving from a Halloween animation).*

## The artistic challenge

So far I've just talked about the technical side of making generators. But all that is worth nothing without good art that people enjoy. The principle artistic lesson I have learned from working on MakeSweet is that I am not an artist. There is an interesting challenge in creating good generators with a "narrative" that users identify with. Here are some of the difficulties of the medium, at least as implemented on MakeSweet:

- The generator must have some flexibility to accommodate unknown user input. You don't control the whole story.
- For animations: you have just a few seconds to tell the story. The longer the animations, the more CPU time burned producing them, and the fewer people will stick around to create them.
- Resolution, frame-rate, and palette are limited (especially if GIF animations are used).

Here are some styles of generator that work:

- Simple scenes based on holidays such as Halloween, Valentine's Day, and so on. These occasions have a lot of shared cultural knowledge to build on.

- Scenes where people already expect to see messages – billboards, monitors, televisions, tattoos, flags, signs, etc.

Sometimes generators fail to attract interest if they are not sufficiently centered on the user input, or don't really have some root in popular culture that helps people grasp them instantly. For example, the heart design in Figure 3 is part of an elaborate animation of falling rain drops that turn out to be heart shaped. It ended with a zoom-in on a drop magnifying the sun, which had the user's image overlaid on it. It was intended for Valentine's day, and wasn't a complete flop, but neither was it a great success. People instead sought out a much simpler design from the previous year, an animation of a locket in the shape of a heart opening to show two photographs. That animation had a much clearer narrative and hook for the user message.

## The community challenge

My hope with MakeSweet is that, by solving the technical problem of making generators, I could support people with actual artistic talent in their creation of generators online, for fun or as part of an interactive portfolio. I believe Blender is a great choice for this. To see why, let's look at the procedure I ended up using to make generators with Blender online:

1 First, I design an interesting .blend that does something neat with an image texture of my son (my standard test image).

2 I make a short configuration file that specifies the image texture (or textures) that should be customizable by the user.

3 I have Blender pre-compute a mapping from an input image (or images) to a render.

4 I upload the mapping to the website. Done!

The first two steps require no special skills beyond a knowledge of Blender. And the remaining steps can be made fully automatic. So there's no reason why anyone who knows Blender shouldn't be able to make generators. And in fact, I created a service on MakeSweet called the Generator Wizard for doing just this. It is in testing right now, and I encourage anyone interested to give it a try (http://makesweet.com/wizard).

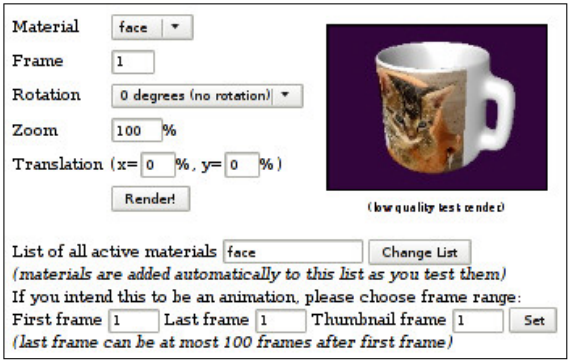


***Figure 4:*** *The Generator Wizard lets you upload a .blend file and convert it into an online generator. For example, here we upload a cup model with an image texture on its surface (where the cat is), and what we get is a webpage that lets anyone replace that texture with their own picture or words. Code is also provided for embedding the generator on other websites.*


by Paul Fitzpatrick

With the wizard, we have entered the world of "generators of generators" – we've made a tool that converts user supplied material (a .blend) into a generator. We can push this idea further, and develop other generators of generators (let's call them "GOGs") with the following trick: the user provides parts that the site assembles into a .blend, and then a generator is made from that .blend as before. So far I've had most success with a "Billboard Generator of Generators." This is a small Flash widget that lets a user select a flat surface

The widget does this by computing the 3D location of the surface and then generating a simple .blend file that does the necessary projection.

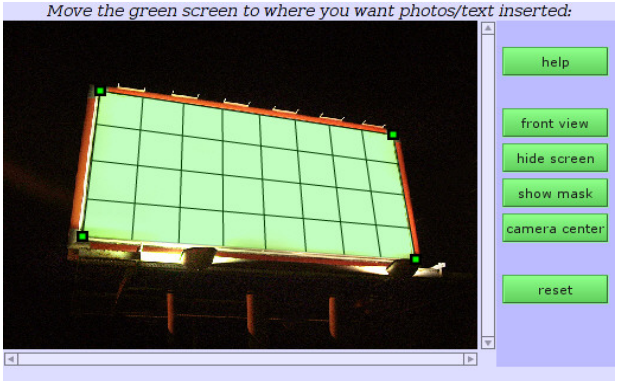


***Figure 5:*** *The Billboard GOG. Users upload a photo of a billboard or any other object with a flat rectangular surface, and then identify the corners. A .blend is then assembled with their photo in the background and the appropriate 3D plane overlaid on it. A generator can then be made.*

Users can advance by learning to apply masks for flat surfaces that are not rectangular (if you look closely, there are several examples in Figure 1). For anything more complicated, they are nudged towards learning Blender. The billboard generator has proven popular, and scenes made with it by users now dominate the site.

## Conclusion

It turns out that Blender is a great file format for expressing visual generators. It nicely separates out the artistic work from the geeky integration. And on that geeky side, Blender is a joy to integrate, and plays very well with other software. MakeSweet was a lot of fun to put together, and has already been a lot more successful than I dared wish for. I hope it will help expose Blender to some of the vast horde of people out there who would love to play with 3D, but haven't yet realized that there's really nothing stopping them anymore. Two years ago, I was one of them.

**Paul Fitzpatrick**

My day job is the RobotCub humanoid robotics project, based in Genoa Italy. Our humanoids have completely open source hardware designs and software.

Website: paul.giszpatrick.com