# Bomber Bug

**VISIT WWW.INFINTYPBR.COM FOR THE LATEST UPDATES, FORUMS, SCRIPT SAMPLES & MORE ASSETS.**



***Please leave a rating & review for us at the Unity Asset Store! If you need support, email us or write us on the forums, but an honest review of our package will help other developers make a decision to purchase this and support us as we make more killer assets.***

# 1. Introduction

"Bomber Bug" is a procedural PBR character designed for video games developers. The procedural aspect means there are virtually unlimited looks you can give to the character, creating a unique look that no one else has. Physically Based Rendering means the look can appear hyper realistic.

Due to all of this — and the physical size of the dragon and it's many optional parts — there is a little setup involved. It shouldn't take long and maybe it'll be quite fun, as you'll get to fine-tune the look of your character.

In most cases the Quick Set Up section will be all that you need. If you're interested in knowing more about each of the values you're able to tweak, check out the Procedural Values section.

For advanced users, if you're interested in scripting run time changes in the texture of the model, refer to the Scripting section.

Finally we include a brief list of the Animations currently included.

We plan on updating our assets periodically, so please check the Asset Store for available updates. *Register your package at www.InfinityPBR.com to get email notifications when we update a package, as well as pre-release and bonus content.*

**** If you have upgraded from a previous version, please check the Change Log at the end of this document to make sure we haven't changed anything you rely on.***

***NOTE: The Bomber Bug was designed to have a semi-transparent "bulb" body. Unfortunately I haven't been able to get this to work as I had hoped, since currently Unity has trouble with multiple transparent objects at once. I will update this as soon as I figure out how to make it work properly.***

***Until that time, you can simply ignore the "Opacity" for the Bug Body, as well as the Emissive values for the Inner Sphere, which isn't visible with an opaque body.***

***The wings, however, are able to be semi-transparent if you so choose. You should make a 2nd "Transparent" material to use on the wings, and keep the opaque version on the rest of the Bug Add On body parts.***

# 2. Quick Set Up

This quick guide will work for most users, and does not allow for run time changes in the look of the textures. *For videos, please visit our website at www.InfinityPBR.com where you will find much more detailed examples.* ***We highly suggest you create your maps in a new project.***

1. Load the Assets/SFBayStudios/SFB Demo Scenes/SFB Bomber Bug Texture Customization.unity scene
2. Select the "BomberBug" Procedural Material in the inspector. Assets/SFBayStudios/SFB Bomber Bug/Procedural Materials/SFBv#_BomberBug_v#
3. Be sure to check the "Enable Texture Modification" box under the top "Main" section. Also make any adjustments to the Environment options, such as Ground Dirt, Damage etc. You only need to adjust the "Body", "Flesh" & "Leg Color", as the next step will copy those changes to the rest of the textures.
4. In the scene view, select "Copy This To That Control". This editor script (see Inspector) will copy the settings from "Body" to all the listed sections in the target materials. You can add or remove them as you'd like, but if you plan on making a custom Copy This To That controller, I suggest you create a new object, and keep the demo defaults as is.
5. Click the "Copy Settings" button, and wait until all materials are finished updating. You'll see them update in the scene view, but it may take a few moments.
6. If you want to make any more modifications to individual parts or materials, do so now. If you click "Copy Settings" again, any changes you make will be lost.
7. When you are satisfied with the look of your model, select the "Mass Exporter" object. In the inspector, confirm that all of the Procedural Materials are seen in the "Substances" array. Give your new character a descriptive name, such as "Blue Bug", and choose whether to remove Emissive & Height maps. If you are going to share maps with a previous export, add the "Previous Group Name" and select which maps to share. If this is your first export, or you've changed so much of the look that you can't share maps, uncheck the boxes. Finally, click "Export Materials." It will take a few minutes for Unity to import all the .tga files that are created.
8. When complete, you'll find your game-ready materials in Assets/SFBayStudios/Exported Materials/[Group Name]
9. Don't forget to choose the correct LOD for your game, and play with the size settings of the textures to optimize their system resource usage.
10.

**What Material Goes Where?**

*There are quite a few sub-meshes that you can optionally use. Some share materials. Refer to the list below.*

| Sub-Mesh | Material |
|---|---|
| Abdomen | Main |
| Body | Main |

| | |
|---|---|
| Carapace | Add On |
| Eyes | Main |
| Head | Main |
| R/L Legs A | Add On |
| R/L Legs B | Main |
| R/L Legs C | Add On |
| R/L  Pincer | Add On |
| R/L Horns | Add On |
| R/L  Wing | Add On |

**** For more in depth instructions, please refer to the videos linked at http://www.InfinityPBR.com***

# 3. Procedural Values

The included Run Time procedural materials are designed to be used in game. They start with "SFB_RT_". You'll find some basic scripting guidelines here, but we encourage you to use the Forums on the Unity website if you have more questions, as we aren't the best coders. *NOTE: Due to the multiple sub-meshes, these would have to be used, and updated in code, per material. However, they should all blend perfectly together.*

*Starting in Unity 5.4 (maybe earlier), if you hover your mouse over a value / name in the inspector, a small tooltip will tell you the "name" of that value, which you'll need for coding changes to the procedural materials.*

**SFB_RT_TextureBlend:** Use this to blend between two textures. It accepts two groups of exported maps, and has a float value to blend between the two. One potential use is to blend between a "clean" and "damaged" version of an object. Perhaps the more it is used, the more the "Damaged" version is shown.

**SFB_RT_DirectionalBlood:** Bring in your ready-to-go texture maps, along with the Positional Map for the object (included in the package). Choose the direction appropriate for your object, and then you can code the material to update in script. This could be used to add blood to an object whenever it's "used", and the blood can fade out over time.

### SFB_RT_TextureBlend

| Category | Name | ID \| Type Min,Max | Description |
|---|---|---|---|
| N/A | Blend Amount | `blendAmount`<br>`float (0.0,1.0)` | Blend amount between Texture group 1 and Texture group 2. |

### SFB_RT_SFX

| Category | Name | ID \| Type Min,Max | Description |
|---|---|---|---|
| | Water Level | `SFXWaterLevel`<br>`float (0.0,1.0)` | How much water |
| | Water Details | `SFXWaterDetails`<br>`float (0.0,1.0)` | Details of the water |
| | Refraction | `SFXRefraction`<br>`float (0.0,1.0)` | Refraction of the water. (Try 0 for an "ooze" look) |
| | Reflection | `SFXReflection`<br>`float (0.0,1.0)` | Reflection amount |
| | Reflection Distance | `SFXReflectionDistance`<br>`float (0.0,1.0)` | Reflection distance |
| | Flow Direction | `SFXFlowDirection`<br>`float (0.0,1.0)` | Changes the direction the water appears to be flowing |

| Category | Name | ID \| Type Min,Max | Description |
|---|---|---|---|
| SFX | Water Color | SFXWaterColor<br>Color | Color of the water (clear = default) |
| | Ice | SFXIce<br>float (0.0,1.0) | How much of the water has turned to ice? Water is required for this to work. |
| | Ice Details | SFXIceDetails<br>float (0.0,1.0) | Details of the ice |
| | Snow | SFXSnow<br>float (0.0,1.0) | Amount of snow |
| | Moss | SFXMoss<br>float (0.0,1.0) | Amount of moss |
| | Moss Scale | SFXMossScale<br>int (1,4) | Scale of the moss texture |
| | Moss Color | SFXMossColor<br>Color | Color of the moss |

## SFB_RT_DirectionalBlood

| Category | Name | ID \| Type Min,Max | Description |
|---|---|---|---|
| N/A | Axis | axis<br>int (1,6) | Which direction should be blood go? X, X-inverted, Y, Y-inverted, Z, Z-inverted |
| Blood | Height | bloodHeight<br>float (0.0,1.0) | How high does the blood extend |
| | Level | bloodLevel<br>float (0.0,1.0) | How “thick” is the blood, works in conjunction with height |
| | Contrast | bloodContrast<br>float (0.0,1.0) | The contrast of the pattern |
| | Color | bloodColor<br>Color() | Color of the blood |
| | Roughness | bloodRoughness<br>float (0.0,1.0) | How reflective the blood is, suggested range 0.25-0.8 |

# 4. Scripting

It's possible to change values during run time. We include a few versions of the material, some of which are optimized for common run-time options. In those cases, you'll likely want to bake maps for the base materials you plan on using (which do not change at run time), and use the optimized versions. This will speed up the changes in game.

*Please Note: We are not the best coders. There may be more ways of doing what we're doing, perhaps better ways. Please use the forums on our site and the Unity forums if you'd like to discuss or ask the community about various ways of doing this.* ***We are also using Unity Script because, simply, it's what we currently understand.*** *Check out our demo scripts for more extensive examples.*

```
var substance                : ProceduralMaterial;

// Set an Int or a Float value
substance.SetProceduralFloat("Grunge2Volume", 0.5);

// Set a Color value
substance.SetProceduralColor("Grunge2Color", Color(1,1,1,1));

// Get a Vector2 value
var currentOffset   : Vector2 = substance.GetProceduralVector("Grunge2Offset");

// Set a Vector2 value
substance.SetProceduralVector("Grunge2Offset", Vector2(currentOffset[0],currentOffset[1]));
```

*Note: Visit http://www.InfinityPBR.com for tutorials and videos showing more things you can do with our work.*

*ALSO: If you haven't checked out the Unity Docs on Procedural Materials, you may wish to do so. There are many things you can do with them that I don't explain here, or even know of myself.*

# 5. Animations & Audio

We've included a sizable selection of animations designed specifically for this character. Read about them here. Many, if not all, can be looped in various ways inside Unity to create animation combinations.

Some animations, marked below, use an Avatar Mask. This allow the Bomber Bug to use ground animations while flying. Please check the Demo Scene & Animator Component & scripts to see how I manage the Body layer, smoothing the weight from 0 to 1 and back to 0 at the appropriate times.

| Animation | Looped? | Description |
|---|---|---|
| Air to Ground | No | Transition from Air to Ground |
| Ground to Air | No | Transition from Ground to Air |
| Attack 1 | No | Pincer Attack |
| Attack 2 | No | Horn Attack |
| Attack 3 | No | Runs forward and hits enemy with the horns |
| Death | No | Death Animation w/o Explosion |
| Explode | No | The Bug Explodes himself |
| Hit Reaction | No | Got Hit Reaction |
| Idle | Yes | Idle Loop |
| Idle Break | No | Break from the Idle loop |
| Jump Back | No | Dodge animation |
| Magic Spit | No | Bug spits something from its mouth |
| Run | Yes | Fast movement forward, like a cockroach |
| Walk | Yes | Walk forward |
| Walk Back | Yes | Cautious walk backward |
| Fly Back | Yes | Cautious fly backward |
| Fly Death | No | Flying to death, no explosion |
| Fly Dodge | No | Dodge while flying |
| Fly Forward | Yes | Movement forward while flying |
| Fly Hit | No | Got Hit animation while flying |
| Fly Idle | Yes | Idle animation in the air |
| Fly Attack 1 | No | [Avatar Mask] Same as Attack 1 |

| Animation | Looped? | Description |
|---|---|---|
| Fly Attack 2 | No | [Avatar Mask] Same as Attack 2 |
| Fly Attack 3 | No | [Avatar Mask] Same as Attack 3 |
| Fly Magic Spit | No | [Avatar Mask] Same as Magic Spit |
| Fly Dodge Alt | No | [Avatar Mask] Ground Dodge used in Flight |
| Fly Idle Break | No | [Avatar Mask] Ground Idle Break used in Flight |
| Fly Got Hit Alt | No | [Avatar Mask] Ground Got Hit used in Flight |

## Audio

All animations now include audio that works with them quite well.  Most of the time audio is triggered via Animation Events, but sometimes scripting is used as well.  Please check the demo scene and dig into the audio controller to see how I accomplish various audio tasks.

# 6. Level Of Details

There are multiple level of details available.  The full resolution is of course the best to use in close up views.  However each of the levels could be better for when the character is further away, or when they're not as visible based on your game.  The default LODGroup may be a good option for many games.

However, it may be a good use of time to pick and choose which LOD to use for your close-up scene, as you may like a lower LOD for the body, but a higher LOD for the spikes.  ** LOD 4 & 5 are kind of the same, as it couldn't get any lower.*

| | |
|---|---|
| LOD 0 | 19892 |
| LOD 1 | 9946 |
| LOD 2 | 5092 |
| LOD 3 | 2605 |
| LOD 4 | 1361 |
| LOD 5 | 1573 |

# 7. Change Log

| | |
|---|---|
| v3 | Includes full audio |
| v2 | Added new “Texture Customization” scene. |
| | Added “Mass Exporter” and “Export Single Material” editor scripts |
| | Added “Copy This To That” Editor Script |
| v1 | Initial Version. |