Dragons Pack

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

1. INTRODUCTION
2. QUICK SET UP
3. PROCEDURAL VALUES
4. SCRIPTING
5. ANIMATIONS
6. LEVEL OF DETAIL
7. CHANGE LOG

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

"Dragon" 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.

**** 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.***

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.***

NOTE: *The Dragons texture maps are separated among different materials, to ensure high resolution textures. The Dragon is large, after all. We've created some helper scripts all titled "SFB_DragonMatch[???]", which will match the texture of one sub-mesh to the texture of the main body. Use these to avoid having to manually adjust each section of the dragon. [Most match the Body, a couple may match other aspects, like the Fins, so check each sub-mesh for specifics. You can turn of matching for individual parts as well, via the Inspector.]*

The prefab Dragon_Demo already has these attached to the parts. ***Therefor, we suggest you use this prefab when creating the look of your dragon!***

1. Bring the model prefab into the Scene view.
2. ***[If the model is pink]*** Find the latest procedural materials: Assets/SFBayStudios/SFB Dragon/Procedural Materials/?????? & drag each one onto each mesh of the model. Refer to the table below to know which model goes on which sub-mesh. It may take a bit of time during any of these steps for Unity to pre-build the material. After dragging, the model may appear black or another solid color until the process is complete. Please be patient. (We hear this speed is based mostly on your GPU, and we've been told that there will be a good speed boost when 5.2 is eventually released.)
3. Rotate the camera in the Game and/or Scene view to something that you like, and select the Procedural Material for the **BODY (SFB_DragonBody)** in the Project view to load it in the Inspector. You can select either the material or it's parent, later you'll need the parent selected, so we encourage you to do that now. *(Rectangle icon, not circle)*
4. Adjust the various aspects of the material to obtain the look you like. Each time you change something Unity may take a moment to rebuild the material. This is not a run-time optimized material, and with so many options, it may take a few moments to complete. Usually once an aspect has been updated once, the subsequent updates will be much snappier. *NOTE: The "DragonMatch" helper scripts will match updates to the Body material after the body has finished updating. So the body will change looks and THEN the other parts will change.*
5. When you are satisfied, with the parent of the Procedural Material selected, choose Window/Save Texture [Command-Shift-T] to save the look you've created as a Standard Shader Material. This process will take a couple minutes, but when it's done, the texture maps will be exported and a new material with those maps will be waiting for you to use in your game. You will need to choose the "EXPORT_HERE" folder when prompted, but otherwise this is an automated function. *You need to do this for each material you want to use in your game. However, if you aren't planning on using one of the optional pieces (like Fins or the Saddle or any of the Heads), you can of course ignore those entirely.*

6. 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. You can make as many versions as you'd like, perhaps allowing for multiple versions of "Dragon" enemies in your game, from simple Dragons to more powerful Dragons.

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
Body	SFB_DragonBody
Bridle A	SFB_DragonSaddle
Bridle B	SFB_DragonSaddle
Bridle C	SFB_DragonSaddle
Carapace	SFB_DragonCarapace
Carapace Saddle	SFB_DragonCarapace
Fins	SFB_DragonFins
Fins Saddle	SFB_DragonFins
Head A	SFB_DragonHeadA
Head B	SFB_DragonHeadB
Head C	SFB_DragonHeadC
Reins A	SFB_DragonSaddle
Reins B C	SFB_DragonSaddle
Saddle	SFB_DragonSaddle
Saddle Carapace	SFB_DragonSaddle
Spikes	SFB_DragonSpikes
Spikes Saddle	SFB_DragonSpikes
Tail A	SFB_DragonTailA
Tail B	SFB_DragonTailB

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

SPECIAL NOTES

There are two potential “bugs” that you may have to deal with.

1. Sometimes after dragging the created Standard Shader materials onto the object, you’ll need to click on the object, and expand the material in the Inspector. Unity seems to not update the material until after you do so.
2. Our automated texture export should put the MetallicRoughness map into the Metallic field, but sometimes it puts the map called “Z_Metallic” instead. If this happens, the object will look funny, and likely be very reflective. You’ll need to manually place the MetallicRoughness map into the Metallic field if this happens. [Unity uses the alpha channel of the Metallic field as the roughness map.]

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.*

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` `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
SFX	Water Level	`SFXWaterLevel` `float (0.0,1.0)`	How much water
	Water Details	`SFXWaterDetails` `float (0.0,1.0)`	Details of the water
	Refraction	`SFXRefraction` `float (0.0,1.0)`	Refraction of the water. (Try 0 for an "ooze" look)
	Reflection	`SFXReflection` `float (0.0,1.0)`	Reflection amount
	Reflection Distance	`SFXReflectionDistance` `float (0.0,1.0)`	Reflection distance
	Flow Direction	`SFXFlowDirection` `float (0.0,1.0)`	Changes the direction the water appears to be flowing
	Water Color	`SFXWaterColor` `Color`	Color of the water (clear = default)
	Ice	`SFXIce` `float (0.0,1.0)`	How much of the water has turned to ice? Water is required for this to work.

Category	Name	ID | Type Min,Max	Description
	Ice Details	SFXIceDetails float (0.0,1.0)	Details of the ice
	Snow	SFXSnow float (0.0,1.0)	Amount of snow
	Moss	SFXMoss float (0.0,1.0)	Amount of moss
	Moss Scale	SFXMossScale int (1,4)	Scale of the moss texture
	Moss Color	SFXMossColor Color	Color of the moss

SFB_RT_DirectionalBlood

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

5. Animations

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.

Animation	Looped?	Description
Attack01	No	Standing Attack #1
Attack02	No	Standing Attack #2
BreatheFire	No	Standing Breathing Fire Forward
Death	No	Death while Standing
FireHead01	No	*[Head Only]* Fire Breathing #1 — Use with Animation Layers
FireHead02	Yes	*[Head Only]* Fire Breathing #2 — Use with Animation Layers
Fly	Yes	Flying Forward
FlyAttack	No	Attacking with Limbs while Flying
FlyBackward	Yes	Flying Backward
FlyBreatheFire	No	Breathing Fire while Flying
FlyDeath01	No	The start of the "Death" loop
FlyDeath02	Yes	The middle part, this loops entirely until you call the end.
FlyDeath03	No	The end of the "Death" loop, falling from the sky.
FlyDive	Yes	Diving Down at a steep angle
FlyFast	Yes	Flying Fast, forward
FlyGlide	Yes	Gliding, wings are not beating
FlyHit	No	Get hit while flying
FlyIdle	Yes	Idle flying in the air, keeping the back up to not make the Rider fall off.
FlyStaticSaddle	Yes	Flying forward, but center is on the saddle, so the rider doesn't actually move up and down with each beat of the wings. This could be useful for VR uses, when the camera is in the saddle position, to help players avoid VR Sickness.
Hit01	No	Got hit while standing #1
Hit02	No	Got hit while standing #2
Idle	Yes	Standing Idle
IdleBreak	No	Standing Idle Break
LookBackL01	No	Start of the "Look Back Left" loop
LookBackL02	Yes	Looped pose where Dragon is looking back @ the Rider

Animation	Looped?	Description
LookBackL03	No	End of the “Look Back Left” loop
LookBackR01	No	Start of the “Look Back Right” loop
LookBackR02	Yes	Looped pose where Dragon is looking back @ the Rider
LookBackR03	No	End of the "Look Back Right” loop
Run	Yes	Dragon running on the ground
TailWhipL	No	Attack where the dragon whips his tail to the Left
TailWhipR	No	Attack where the dragon whips his tail to the Right
Walk	Yes	Dragon walking forward
WalkBackward	Yes	Dragon walking backward

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.

	LOD0	LOD1	LOD2	LOD3	LOD4	LOD5
Body	19502	13840	7579	4043	2449	1527
Bridle A	2562	1415	744	292	82	19
Bridle B	2668	1403	635	237	64	24
Bridle C	1842	880	413	166	43	7
Carapace	6336	4027	1848	819	308	131
CarapaceAlt	5382	3253	1498	684	268	123
Fins	8736	4450	2326	1072	530	276
FinsAlt	7966	4132	2134	974	504	228
Head A	12912	3333	1370	651	378	220
Head B	12938	5374	2458	1221	618	358
Head C	13916	4551	1974	940	488	284
Reins A	1640	766	430	300	174	58
Reins B C	1640	760	414	290	144	68
Saddle	6348	4148	2066	1090	497	199
Saddle Alt	6348	4248	2107	1116	527	214
Spikes	9328	3007	1402	663	222	6
Spikes Alt	8096	2661	1212	576	189	12
Tail A	1712	1240	638	350	211	132
Tail B	4250	3573	2282	1281	685	303

7. Change Log

v26	Adjusted some of the default maps
v24	Initial Version.