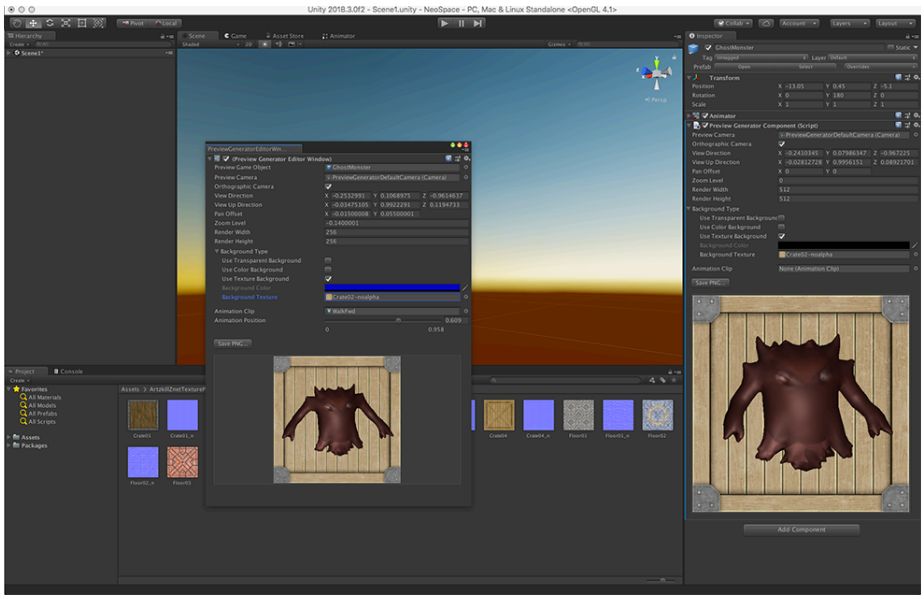
Stellar Giant Logo

Welcome to the PreviewGenerator Documentation!

A Generator for thumbnails/previews/icons/sprites from 3D models in Unity. It has a Custom Editor for generating the texture and an in-game component that allows you to create the sprites from a RenderTexture.

This is useful for many things including generating icons at run-time.

Rotate View Direction, Pan and Zoom Camera for precise placement using the mouse.



UnityPreviewGenerator

MIT License - (https://github.com/klhurley/UnityPreviewGenerator/blob/master/LICENSE)

Installation

Installation is pretty straight forward.

You can either clone this project from this repo or find it on the Unity Store for Free at https://assetstore.unity.com/packages/slug/138077

Make sure you use –recurse-submodules to get the submodule if cloning this project ex:

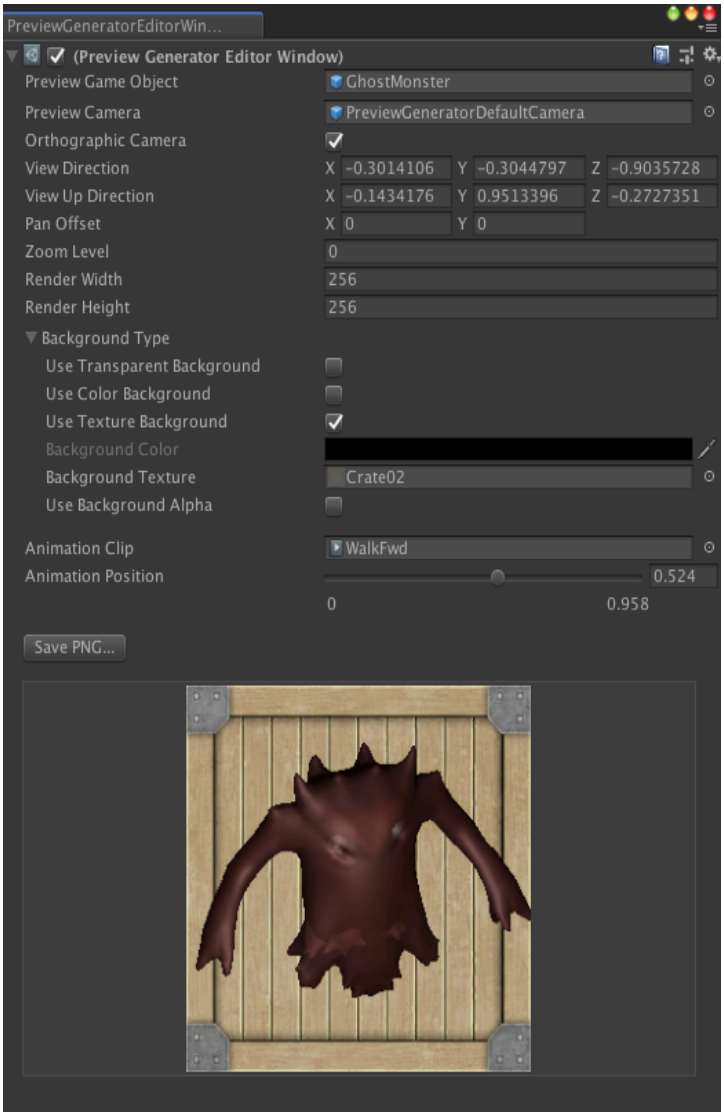
```
git clone --recurse-submodules https://github.com/klhurley/UnityPreviewGenerator.git
```

Editor Window

The pop-up window and properties of the Editor Window can be accessed via the Windows->Preview Generator menu item in Unity.

The Editor Window saves settings under the ProjectSettings diretory of the project as PreviewGeneratorSettings.json.

The settings in both the Editor Window and the Preview Component are exactly the same. Your can access the documentation for the Settings on the Property Settings page.



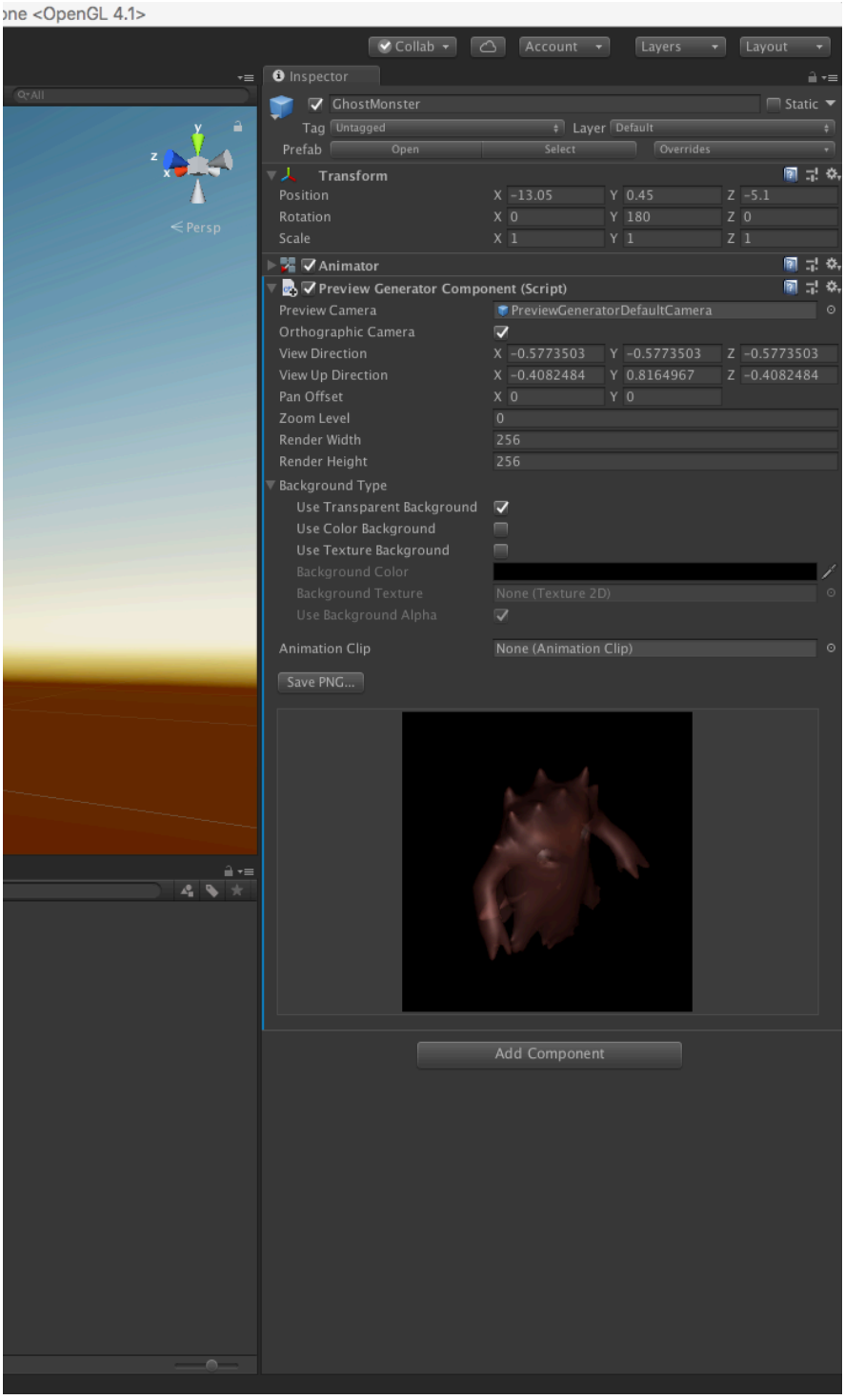
Editor Window Image

Preview Component

The Preview Generator Component is a script (PreviewGeneratorComponent.cs) that uses the same code base as th Editor Window to store information in a prefab or GameObject in the scene.

This data is serialized into the GameObject or Prefab and you can attach the component to any GameObject, i.e. Sprite Object, 3D Model Object, Prefab, etc.

Your can access the documentation for the Settings on the Property Settings page.



Editor Window Image

Camera Manipulation

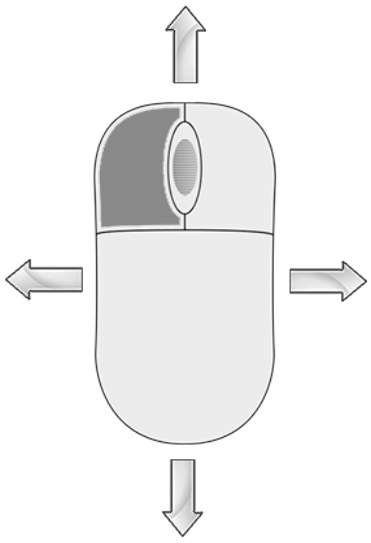
You can rotate the camera around the 3D Model by using the following mouse and keyboard combinations to rotate, pan and zoom.

First place your mouse cursor over the rendered preview image.

Rendered Preview Image

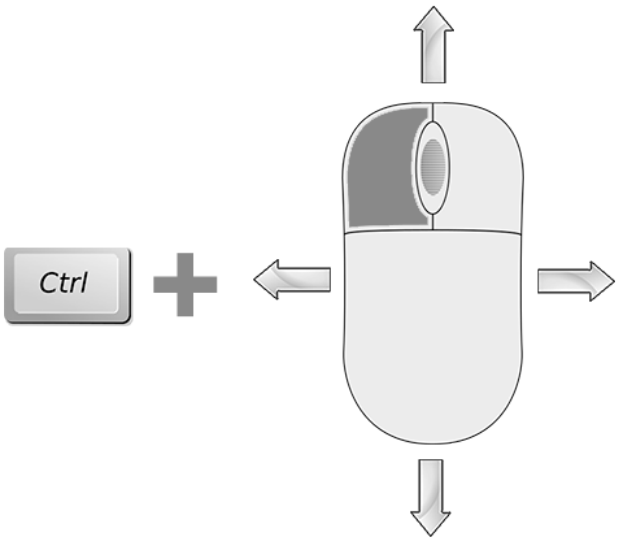
Rotation: To rotate the camera around the object in an ArcBall implementation.

Click and hold the left mouse button and then move your mouse up/down to pitch around the model (rotation around the X-Axis). Right/left mouse movement will yaw around the model (rotation around the Y-Axis).



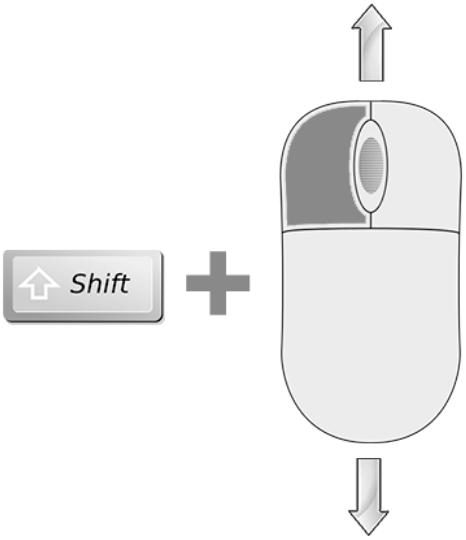
Mouse Move Left Image

Panning: To pan the camera right/left and up/down hold the Ctrl (Control) key on the keyboard and press the left mouse button. Moving the mouse now pans the camera in the X/Y Screen space.



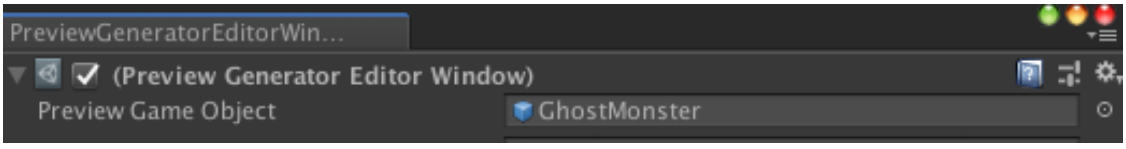
Ctrl Mouse Move Image

Zooming: This is a little trickier but was used in case there is no mouse-wheel on the mouse. To Zoom, simply hold the Shift key on the keyboard and press the left mouse button. Moving up and down will zoom the camer into the image.



Shift Mouse Move Image

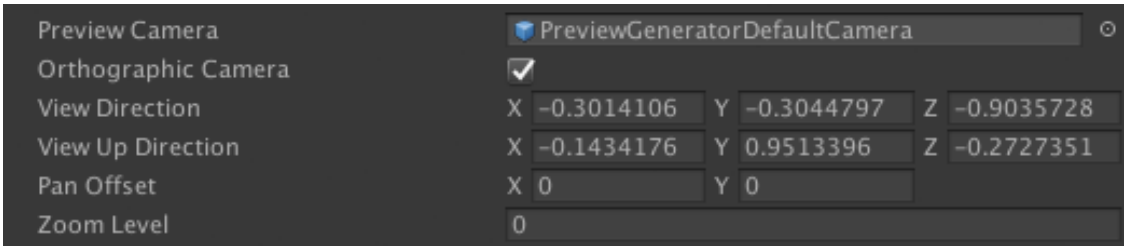
GameObject Target Settings



GameObject Target Settings Image

The **Preview Game Object** is the 3D Model that will be rendered in an animation pose. The prefab or GameObject in the scene should have Render Meshes in it to be rendered.

Camera Settings



Camera Settings Image

The **Camera Settings** include **Preview Camera** which is a prefab that must include a Camera Component.

If the prefab does not contain a camera component the setting will be set to the default camera included under the resources directory. It the Preview Camera is set to 'None' The preview generator will use an internal camera object.

There is also another sample Camera prefab that contains an attached PostProcessing Stack where you can render the icon with PostProcessing. **Hint:** If you render a sprite with the texture and the PostProcessing is attached to you in game Camera, you will get double post processing effects.

Orthographic Camera is to set the PreviewCamera Object to orthographic mode (the default). If you want to use Perspective mode just untick this checkbox.

Camera Manipulation via Keyboard input

See the Camera Manipulation page for how to move the camera with the mouse.

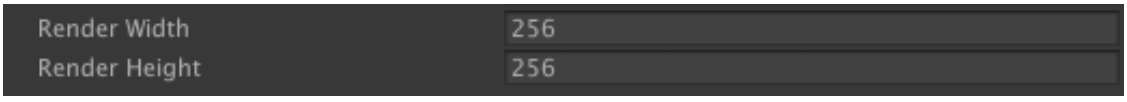
View Direction allows you to set the camera viewing angle with the keyboard. The object in the texture does not actually rotate, the camera rotates around the object using and ArcBall implementation.

View Up Direction allows you to set the camera's up direction with the keyboard. This allows you to view an object in 360 degree view.

Pan Offset allows you to offset the view right/left and up/down with the keyboard. This pans the camera view.

Zoom Level allows you to set the zoom level of the camera to zoom in and out on the GameObject being rendered. The orthographic zoom level is capped at 1.0 zoom up level as the image will zoom out after that threshold.

Rendered Texture Dimensions



Render Dimensions Image

Render Width is the render texture width size that is rendered in memory. (See 'The Rendered Texture' for example)

Render Height is the render texture height size that is rendered in memory

The Rendered Texture

Rendered Texture Image

This is a sample of a texture that has been rendered with a background. The ghost monster is a 3D model I commissioned for one of my games.

The Camera Manipulation page documents how to use the mouse to manipulating the Camera Preview Settings

Background Type and Settings

Background Type and Settings Image

This section allows you to set the type of background you want to render. The options are:

Use Transparent Background - This options will set the alpha channel to all 0.0 (black) where there is no rendered pixels or 1.0 (white) where there is a rendered pixel.

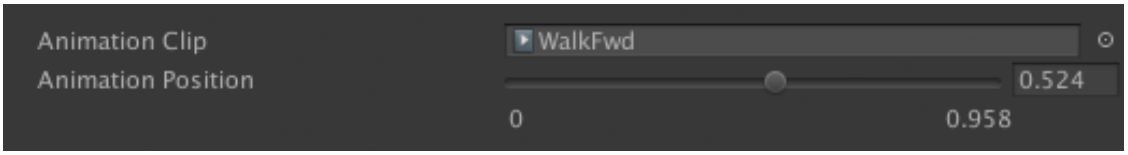
Use Color Background - This option will render a color in the background including alpha.

Background Color allows you to select the background color you would like to render into the background.

Use Teture Background - This options will use a texture.

Use Background Alpha is enabled when a texture is selected. This allows the alpha of the texture to be alpha blended into the background (ex: transparent textures)

Animations Settings



Animation Settings

There are two **Animation Setting** that allow you to set a pose for your 3D Model.

Animation Clip allows you to select an animation clip to use for the render.

Animation Position is a slider bar that allows you to set the time inside the animation. The left value is the minimum animation time (always 0) and the right number below the slider is the max time of the animation clipe. The box to the right of the slide is to set the animation pose time with the keyboard.

When changing the **Animation Position** your render will update and you can slide through your animation in real time. ### SpriteRenderTexture.cs Details

This class is setup to enable runtime sprite generation from 3D Model. The class script is added to a GameObject and is the link between the runtime generator and the Sprite Renderer class

```
public class SpriteRenderTexture : MonoBehaviour
{
    private SpriteRenderer _spriteRenderer;
    private PreviewGeneratorComponent _previewGeneratorComponent;
    private Texture2D _curTexture = null;
    private GameObject _curGameObject;
```

_spriteRenderer is the sprite component that is required and does the rendering of the sprite with RenderTexture attached.

_previewGenereratorComponent is the preview generator component that takes a 3D model with optional background and Postprocessing to render a texture to be used for the sprite.

_curTexture is the current rendered texture from the Preview Generator Component

_curGameObject is the target object to render for the sprite

```
    // Start is called before the first frame update
    void Start()
    {

        _spriteRenderer = gameObject.GetComponent<SpriteRenderer>();
        _previewGeneratorComponent =
gameObject.GetComponent<PreviewGeneratorComponent>();
        if (_previewGeneratorComponent != null)
        {
            SetSprite(true);
```

```
            _curGameObject =
_previewGeneratorComponent.PreviewGenerator.GameObjectToRender;
        }
    }
```

The Start function simple gets the two necessary components that this class acts as the bridge between. It gets the SpriteRenderer component and the PreviewGeneratorComponent.

Next is calls the SetSprite function that grabs the Rendered Texture and creates a new Sprite, see code below in SetSprite.

```
    // Update is called once per frame
    void Update()
    {
        if (_previewGeneratorComponent != null)
        {
            // only gets new renderTexture if object changes or texture changes
            SetSprite(_curGameObject !=
                _previewGeneratorComponent.PreviewGenerator.GameObjectToRender);
        }
    }
```

The update function simple checks if the _curGameObject has changed and Sets the new RenderTexture in a new Sprite. This is useful since this class executes in the Editor as well.

```
   void SetSprite(bool doRender)
    {
        if (_previewGeneratorComponent != null)
        {
            if (_spriteRenderer != null)
            {
                Texture2D spriteTexture =
_previewGeneratorComponent.GetRenderTexture(doRender);
                if ((spriteTexture != null) && (spriteTexture != _curTexture))
                {
                    _curTexture = spriteTexture;
                    Sprite sprite = Sprite.Create(spriteTexture,
                        new Rect(0.0f, 0.0f, spriteTexture.width, spriteTexture.height),
                        new Vector2(.5f, .5f), 100.0f);
                    _spriteRenderer.sprite = sprite;
                }
            }
        }

    }
```

This function takes a bool that will Re-Render the texture (in case of an GameObject change). It does a few safety checks to make sure the components were found and then gets the Rendered Texture from the preview generator. The preview generator only executes if the GameObject being rendered is changed for the underlying rendered texture changes.