```
yEngine;
yEngine.Rendering;
yEngine.Rendering.Universal;
3D_GalaxyMap;
3D_Core;
3D_Combat;

Assets.Script



t Renderer Feature
--------------------------------------
ed on the Blit from the UniversalRende
ps://github.com/Unity-Technologies/Uni

ended to allow for :
pecific access to selecting a source a
utomatic switching to using _AfterPost
etting a _InverseView matrix (cameraTo
  e.g. Reconstruct world pos from dept
URP v10) Enabling generation of DepthN
  This will only include shaders who h
  (workaround for Unlit Shaders / Grap
--------------------------------------
anilux

eAssetMenu(menuName = "Feature/Blit")]
 class Blit : ScriptableRendererFeatur


blic class BlitPass : ScriptableRender


  public Material blitMaterial = null;
  public FilterMode filterMode { get;

  private BlitSettings settings;

  private RTHandle source { get; set;
  private RTHandle destination { get;

  RTHandle m_TemporaryColorTexture;
  RTHandle m_DestinationTexture;
  string m_ProfilerTag;

  public BlitPass(RenderPassEvent rend
  {
      this.renderPassEvent = renderPas
      this.settings = settings;
```

```
yEngine;
yEngine.Rendering;
yEngine.Rendering.Universal;
3D_GalaxyMap;
3D_Core;
3D_Combat;

Assets.Script



t Renderer Feature
--------------------------------------
ed on the Blit from the UniversalRende
ps://github.com/Unity-Technologies/Uni

ended to allow for :
pecific access to selecting a source a
utomatic switching to using _AfterPost
etting a _InverseView matrix (cameraTo
  e.g. Reconstruct world pos from dept
URP v10) Enabling generation of DepthN
  This will only include shaders who h
  (workaround for Unlit Shaders / Grap
--------------------------------------
anilux

eAssetMenu(menuName = "Feature/Blit")]
 class Blit : ScriptableRendererFeatur


blic class BlitPass : ScriptableRender


  public Material blitMaterial = null;
  public FilterMode filterMode { get;

  private BlitSettings settings;

  private RTHandle source { get; set;
  private RTHandle destination { get;

  RTHandle m_TemporaryColorTexture;
  RTHandle m_DestinationTexture;
  string m_ProfilerTag;

  public BlitPass(RenderPassEvent rend
  {
      this.renderPassEvent = renderPas
      this.settings = settings;
      blitMaterial = settings.blitMate
      m_ProfilerTag = tag;
      //m_TemporaryColorTexture.Init("
      //if (settings.dstType == Target
      //{
      //  m_DestinationTexture.Init(se
      //}
  }

  public void Setup(RTHandle source, R
  {
      this.source = source;
      this.destination = destination;

2020_1_OR_NEWER
      if (settings.requireDepthNormals
          ConfigureInput(ScriptableRen

  }

  public override void Execute(Scripta
  {
      CommandBuffer cmd = CommandBuffe

      RenderTextureDescriptor opaqueDe
      opaqueDesc.depthBufferBits = 0;

      if (settings.setInverseViewMatri
      {
          Shader.SetGlobalMatrix("_Inv
      }

      if (settings.dstType == Target.T
      {
          if (settings.overrideGraphic
          {
              opaqueDesc.graphicsForma
          }
          cmd.GetTemporaryRT(0, 1, 1,
              );
      }

      //Debug.Log($"src = {source},
      // Can't read and write to same
      if (source == destination || (se
      {
          cmd.GetTemporaryRT(0, 1, 1,
          Blit(cmd, source, destinatio
          Blit(cmd, source, destinatio
      }
```

```
      blitMaterial = settings.blitMate
      m_ProfilerTag = tag;
      //m_TemporaryColorTexture.Init("
      //if (settings.dstType == Target
      //{
      //  m_DestinationTexture.Init(se
      //}
  }

  public void Setup(RTHandle source, R
  {
      this.source = source;
      this.destination = destination;

2020_1_OR_NEWER
      if (settings.requireDepthNormals
          ConfigureInput(ScriptableRen

  }

  public override void Execute(Scripta
  {
      CommandBuffer cmd = CommandBuffe

      RenderTextureDescriptor opaqueDe
      opaqueDesc.depthBufferBits = 0;

      if (settings.setInverseViewMatri
      {
          Shader.SetGlobalMatrix("_Inv
      }

      if (settings.dstType == Target.T
      {
          if (settings.overrideGraphic
          {
              opaqueDesc.graphicsForma
          }
          cmd.GetTemporaryRT(m_Destina

      }

      //Debug.Log($"src = {source},
      // Can't read and write to same
      if (source == destination || (se
      {
          cmd.GetTemporaryRT(m_Tempora
          Blit(cmd, source, destinatio
          //Blit(cmd, m_TemporaryColor
      }
      else
      {
          Blit(cmd, source, destinatio
      }

      context.ExecuteCommandBuffer(cmd
      CommandBufferPool.Release(cmd);
  }

  public override void FrameCleanup(Co
  {
      if (settings.dstType == Target.T
      {
          cmd.ReleaseTemporaryRT(0);
      }
      if (source == destination || (se
      {
          cmd.ReleaseTemporaryRT(0);
      }
  }


ystem.Serializable]
blic class BlitSettings

  public RenderPassEvent Event = Rende

  public Material blitMaterial = null;
  public int blitMaterialPassIndex = 0
  public bool setInverseViewMatrix = f
  public bool requireDepthNormals = fa

  public Target srcType = Target.Camer
  public string srcTextureId = "_Camer
  public RenderTexture srcTextureObjec

  public Target dstType = Target.Camer
  public string dstTextureId = "_BlitP
  public RenderTexture dstTextureObjec

  public bool overrideGraphicsFormat =
  public UnityEngine.Experimental.Rend


blic enum Target

  CameraColor,
  TextureID,
  RenderTextureObject
```

```
      else
      {
          Blit(cmd, source, destinatio
      }

      context.ExecuteCommandBuffer(cmd
      CommandBufferPool.Release(cmd);
  }

  public override void FrameCleanup(Co
  {
      if (settings.dstType == Target.T
      {
          cmd.ReleaseTemporaryRT(m_Des
      }
      if (source == destination || (se
      {
          cmd.ReleaseTemporaryRT(m_Tem
      }
  }


ystem.Serializable]
blic class BlitSettings

  public RenderPassEvent Event = Rende

  public Material blitMaterial = null;
  public int blitMaterialPassIndex = 0
  public bool setInverseViewMatrix = f
  public bool requireDepthNormals = fa

  public Target srcType = Target.Camer
  public string srcTextureId = "_Camer
  public RenderTexture srcTextureObjec

  public Target dstType = Target.Camer
  public string dstTextureId = "_BlitP
  public RenderTexture dstTextureObjec

  public bool overrideGraphicsFormat =
  public UnityEngine.Experimental.Rend


blic enum Target

  CameraColor,
  TextureID,
  RenderTextureObject
```

```
blic BlitSettings settings = new();

blic BlitPass blitPass;

ivate RTHandle srcIdentifier, dstIdent

blic override void Create()

  var passIndex = settings.blitMateria
  settings.blitMaterialPassIndex = Mat
  blitPass = new BlitPass(settings.Eve

  if (settings.Event == RenderPassEven
  {
      Debug.LogWarning("Note that the
  }

  if (settings.graphicsFormat == Unity
  {
      settings.graphicsFormat = System
  }

  //UpdateSrcIdentifier();
  //UpdateDstIdentifier();


private void UpdateSrcIdentifier()
{
  srcIdentifier = UpdateIdentifier(set
}

private void UpdateDstIdentifier()
{
  dstIdentifier = UpdateIdentifier(set
}

private RTHandle UpdateIdentifier(Targ
{
  if (type == Target.RenderTextureObje
  {
      return obj;
  }
  else if (type == Target.TextureID)
  {
      m_RTHandle.Init(s);
      return new RTHandle(this, type);
      //return s;
  }
  return new RTHandle();
```

```
blic BlitSettings settings = new BlitS

blic BlitPass blitPass;

ivate RTHandle srcIdentifier, dstIdent

blic override void Create()

  var passIndex = settings.blitMateria
  settings.blitMaterialPassIndex = Mat
  blitPass = new BlitPass(settings.Eve

  if (settings.Event == RenderPassEven
  {
      Debug.LogWarning("Note that the
  }

  if (settings.graphicsFormat == Unity
  {
      settings.graphicsFormat = System
  }

  //UpdateSrcIdentifier();
  //UpdateDstIdentifier();


private void UpdateSrcIdentifier()
{
  srcIdentifier = UpdateIdentifier(set
}

private void UpdateDstIdentifier()
{
  dstIdentifier = UpdateIdentifier(set
}

ivate RenderTargetIdentifier UpdateIde

  if (type == Target.RenderTextureObje
  {
      return obj;
  }
  else if (type == Target.TextureID)
  {
      //RenderTargetHandle m_RTHandle
      //m_RTHandle.Init(s);
      //return m_RTHandle.Identifier()
      return s;
  }
```

```
}
RTHandle;

blic override void AddRenderPasses(Scr


  if (settings.blitMaterial == null)
  {
      Debug.LogWarningFormat("Missing
      return;
  }

  if (settings.Event == RenderPassEven
  {
  }
  else if (settings.Event == RenderPas
  {
      // If event is AfterRendering, a
      if (settings.srcType == Target.C
      {
          settings.srcType = Target.Te
          settings.srcTextureId = "_Af
          //UpdateSrcIdentifier();
      }
      if (settings.dstType == Target.C
      {
          settings.dstType = Target.Te
          settings.dstTextureId = "_Af
          //UpdateDstIdentifier();
      }
  }
  else
  {
      // If src/dst is using _AfterPos
      if (settings.srcType == Target.T
      {
          settings.srcType = Target.Ca
          settings.srcTextureId = "";
          //UpdateSrcIdentifier();
      }
      if (settings.dstType == Target.T
      {
          settings.dstType = Target.Ca
          settings.dstTextureId = "";
          //UpdateDstIdentifier();
      }
  }

  var src = (settings.srcType == Targe
  var dest = (settings.dstType == Targ
```

```
  return new RenderTargetIdentifier();


blic override void AddRenderPasses(Scr


  if (settings.blitMaterial == null)
  {
      Debug.LogWarningFormat("Missing
      return;
  }

  if (settings.Event == RenderPassEven
  {
  }
  else if (settings.Event == RenderPas
  {
      // If event is AfterRendering, a
      if (settings.srcType == Target.C
      {
          settings.srcType = Target.Te
          settings.srcTextureId = "_Af
          //UpdateSrcIdentifier();
      }
      if (settings.dstType == Target.C
      {
          settings.dstType = Target.Te
          settings.dstTextureId = "_Af
          //UpdateDstIdentifier();
      }
  }
  else
  {
      // If src/dst is using _AfterPos
      if (settings.srcType == Target.T
      {
          settings.srcType = Target.Ca
          settings.srcTextureId = "";
          //UpdateSrcIdentifier();
      }
      if (settings.dstType == Target.T
      {
          settings.dstType = Target.Ca
          settings.dstTextureId = "";
          //UpdateDstIdentifier();
      }
  }

  var src = (settings.srcType == Targe
  var dest = (settings.dstType == Targ
  blitPass.Setup(src, dest);             blitPass.Setup(src, dest);
  renderer.EnqueuePass(blitPass);        renderer.EnqueuePass(blitPass);
```