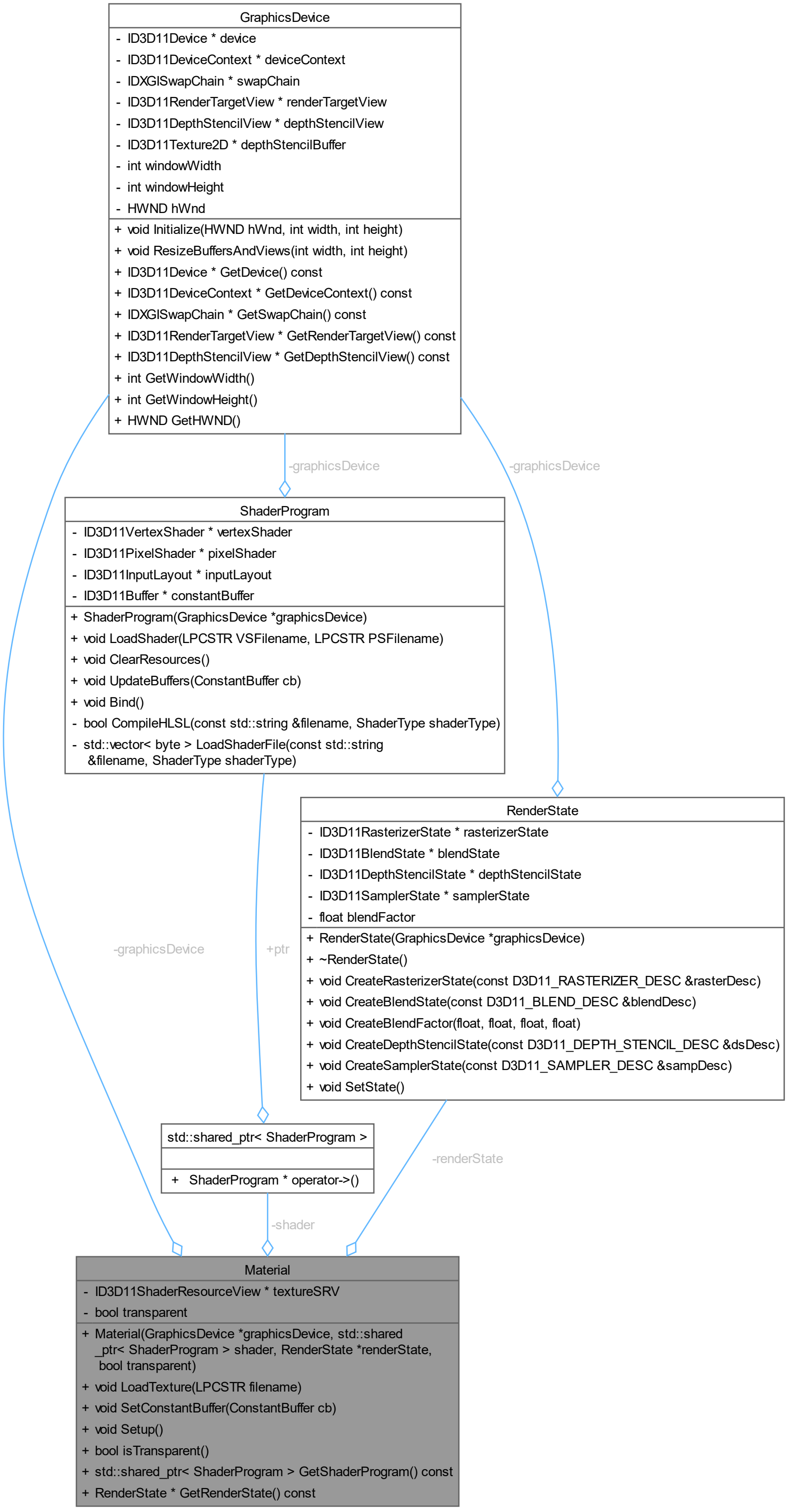
GraphicsDevice
- ID3D11Device * device
- ID3D11DeviceContext * deviceContext
- IDXGISwapChain * swapChain
- ID3D11RenderTargetView * renderTargetView
- ID3D11DepthStencilView * depthStencilView
- ID3D11Texture2D * depthStencilBuffer
- int windowWidth
- int windowHeight
- HWND hWnd
+ void Initialize(HWND hWnd, int width, int height)
+ void ResizeBuffersAndViews(int width, int height)
+ ID3D11Device * GetDevice() const
+ ID3D11DeviceContext * GetDeviceContext() const
+ IDXGISwapChain * GetSwapChain() const
+ ID3D11RenderTargetView * GetRenderTargetView() const
+ ID3D11DepthStencilView * GetDepthStencilView() const
+ int GetWindowWidth()
+ int GetWindowHeight()
+ HWND GetHWND()
-graphicsDevice
-graphicsDevice
ShaderProgram
- ID3D11VertexShader * vertexShader
- ID3D11PixelShader * pixelShader
- ID3D11InputLayout * inputLayout
- ID3D11Buffer * constantBuffer
+ ShaderProgram(GraphicsDevice *graphicsDevice)
+ void LoadShader(LPCSTR VSFilename, LPCSTR PSFilename)
+ void ClearResources()
+ void UpdateBuffers(ConstantBuffer cb)
+ void Bind()
- bool CompileHLSL(const std::string &filename, ShaderType shaderType)
- std::vector< byte > LoadShaderFile(const std::string &filename, ShaderType shaderType)
RenderState
- ID3D11RasterizerState * rasterizerState
- ID3D11BlendState * blendState
- ID3D11DepthStencilState * depthStencilState
- ID3D11SamplerState * samplerState
- float blendFactor
+ RenderState(GraphicsDevice *graphicsDevice)
+ ~RenderState()
+ void CreateRasterizerState(const D3D11_RASTERIZER_DESC &rasterDesc)
+ void CreateBlendState(const D3D11_BLEND_DESC &blendDesc)
+ void CreateBlendFactor(float, float, float, float)
+ void CreateDepthStencilState(const D3D11_DEPTH_STENCIL_DESC &dsDesc)
+ void CreateSamplerState(const D3D11_SAMPLER_DESC &sampDesc)
+ void SetState()
-graphicsDevice
+ptr
std::shared_ptr< ShaderProgram >
+ ShaderProgram * operator->()
-renderState
-shader
Material
- ID3D11ShaderResourceView * textureSRV
- bool transparent
+ Material(GraphicsDevice *graphicsDevice, std::shared_ptr< ShaderProgram > shader, RenderState *renderState, bool transparent)
+ void LoadTexture(LPCSTR filename)
+ void SetConstantBuffer(ConstantBuffer cb)
+ void Setup()
+ bool isTransparent()
+ std::shared_ptr< ShaderProgram > GetShaderProgram() const
+ RenderState * GetRenderState() const