

# C#/.Net-Entwicklung heute am Beispiel einer 3D-Grafik-Engine



### Kontakt

E-Mail:

[roland.koenig@rolandk.de](mailto:roland.koenig@rolandk.de)

Homepage:

[www.rolandk.de/wp](http://www.rolandk.de/wp)

### Berufliches

Schwerpunkt:

Produktentwicklung

Arbeitgeber:

IGZ Logistics + IT, Falkenberg

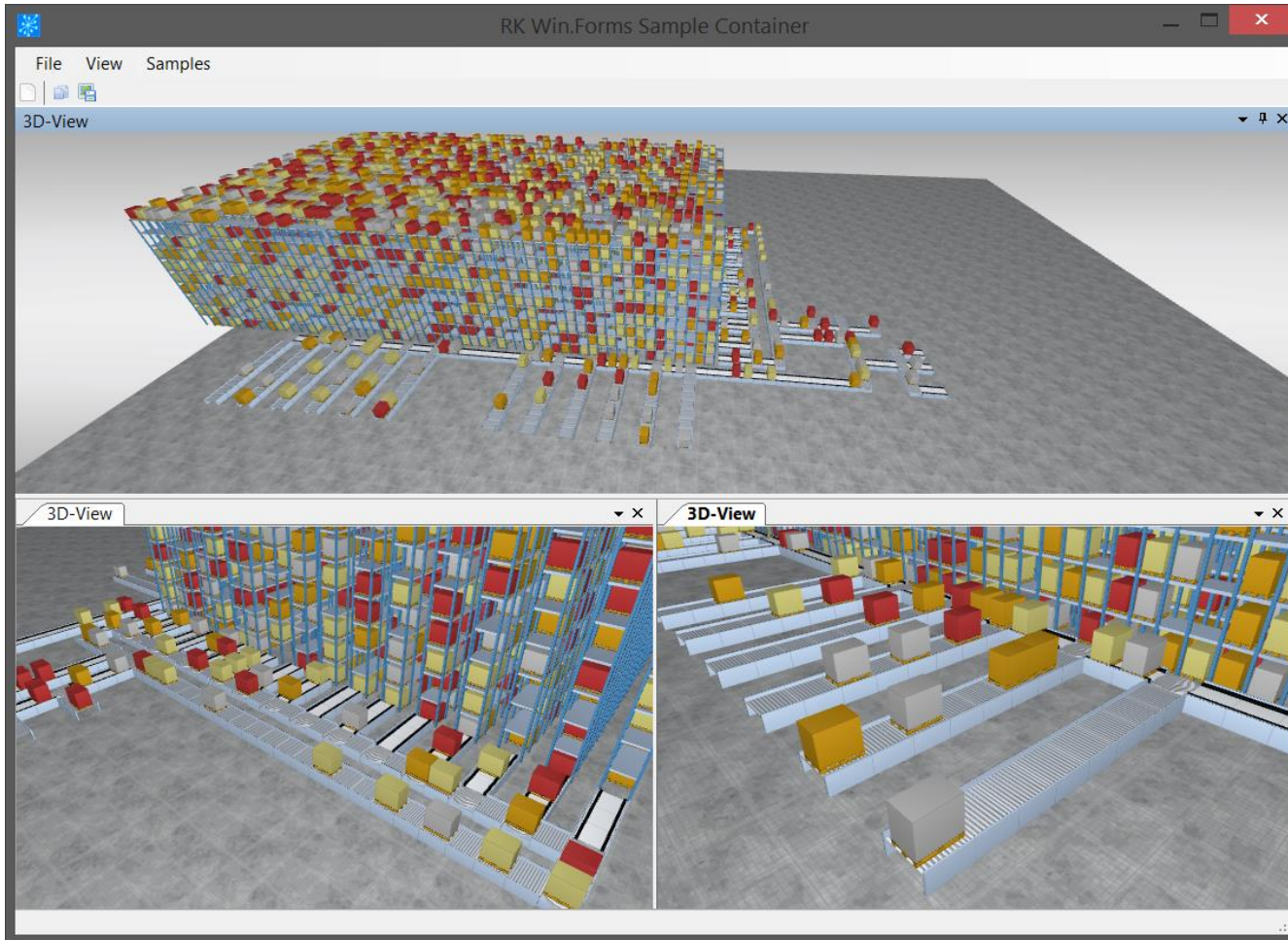


- **Funktionsübersicht 3D-Engine**
- Diverse Aspekte der Entwicklung
  - Verschiedene Plattformen
  - Multi-Core Entwicklung
  - Native Schnittstellen
  - Performance- und Speicher-Analyse
- Best practices...



# Funktionsübersicht 3D-Engine

## Beispiel





■ Zeit pro Frame (Bild): **ca. 33ms**  
(bei 30 FPS)

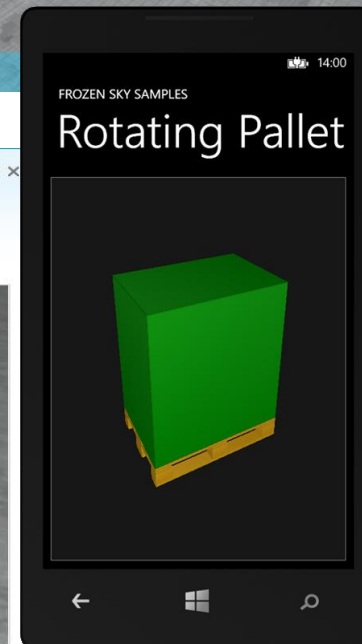
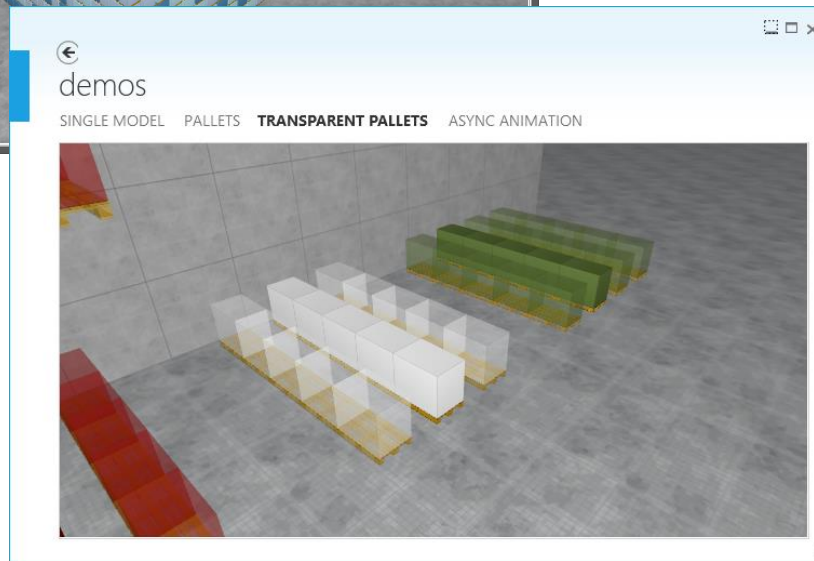
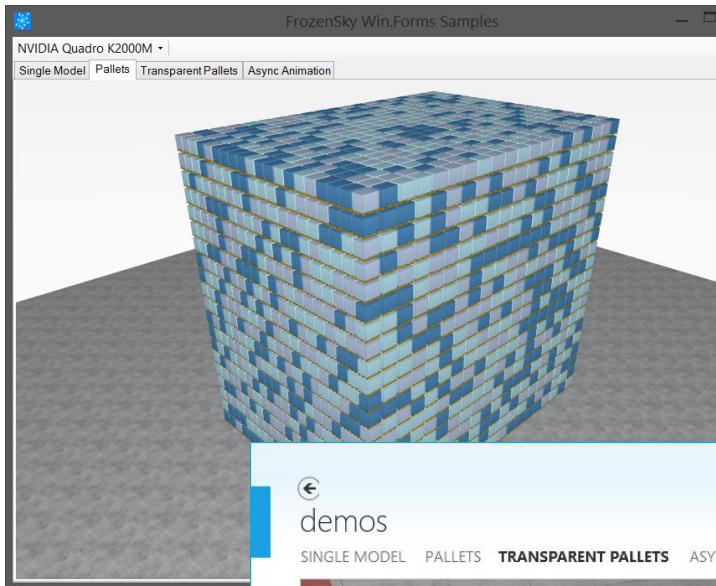
■ Aufgaben pro Frame

- Rendering
- Div. Optimierungen (z. B. Render-Reihenfolge)
- Hit-Testing (Mouse-Over)
- Culling (Sichtbarkeits-Prüfungen)
- Animationen
- Kollisionsprüfung
- Ressourcen laden / entladen
- Gui-Synchronisierung
- Uvm...



# Funktionsübersicht 3D-Engine

## Plattformen



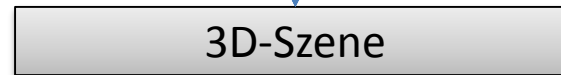
# Funktionsübersicht 3D-Engine

## Hardware-Anbindung

Views:



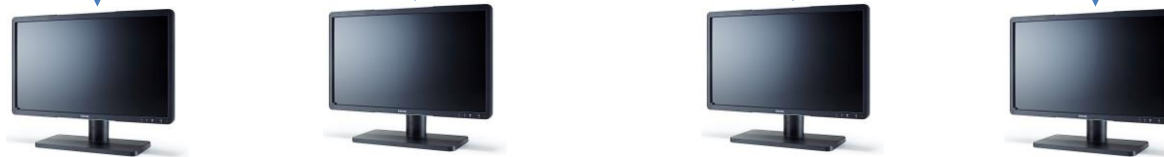
Scene:



Devices:



Outputs:



- Funktionsübersicht 3D-Engine
- Diverse Aspekte der Entwicklung
  - **Verschiedene Plattformen**
  - Multi-Core Entwicklung
  - Native Schnittstellen
  - Performance- und Speicher-Analyse
- Best practices...





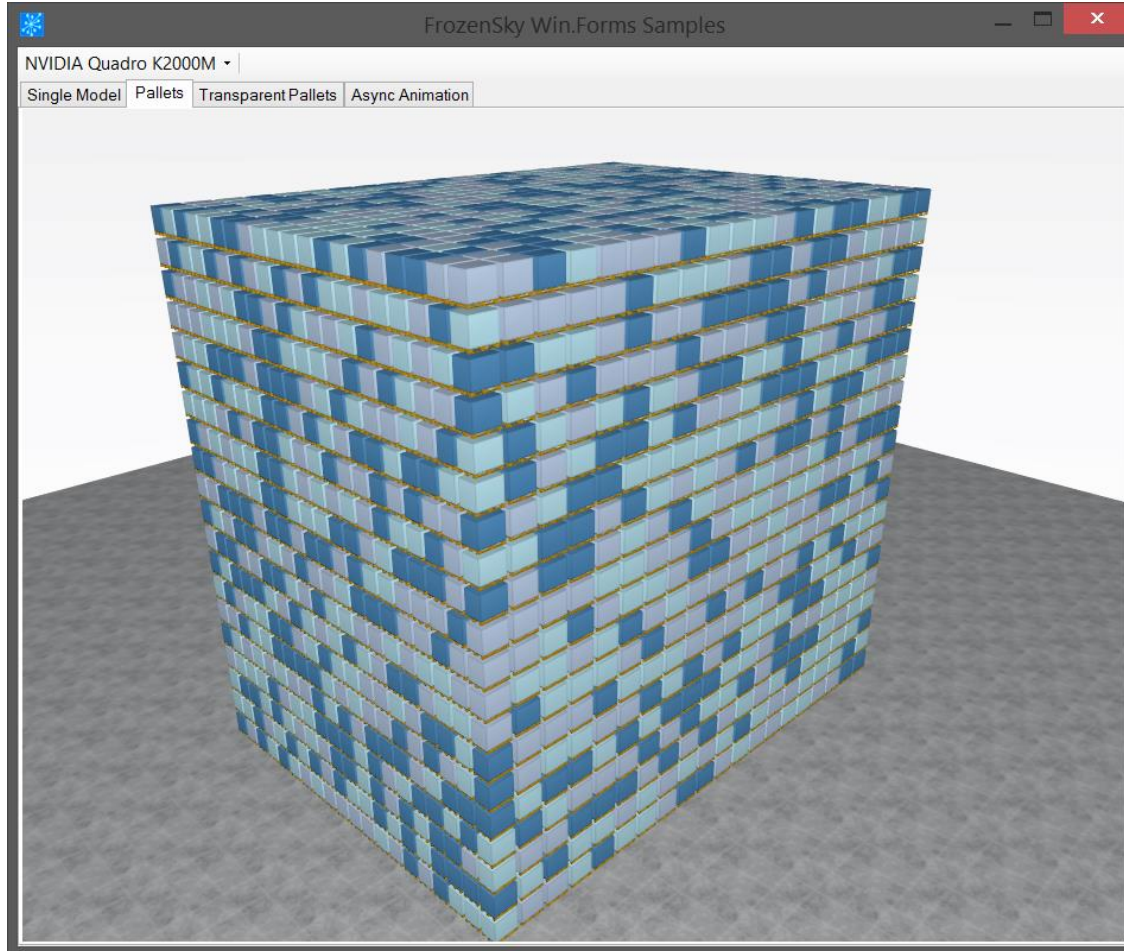
# Verschiedene Plattformen

## Beispiel WPF



# Verschiedene Plattformen

## Beispiel Win.Forms



# Verschiedene Plattformen

## Beispiel WinRT





- Portable Class Libraries nicht wirklich praxistauglich  
(zumindest nicht im gezeigten Beispiel)
- Gemeinsame Quellcode-Dateien, #if...
- WPF bei Integration von Direct3D am aufwändigsten
- Überraschend wenige Konfliktpunkte zwischen WinRT und Desktop  
(Aber: Gui-Schicht muss separat gemacht werden)
  - Größte Umgewöhnung: Keine Thread-Klasse in WinRT
- Windows Phone dafür mehr Unterschiede...



- Funktionsübersicht 3D-Engine
- Diverse Aspekte der Entwicklung
  - Verschiedene Plattformen
  - **Multi-Core Entwicklung**
  - Native Schnittstellen
  - Performance- und Speicher-Analyse
- Best practices...





### ■ Manipulation der Scene

```
await scene.ManipulateSceneAsync((manipulator) =>
{
    // Create floor
    manipulator.BuildStandardConveyorFloor(Scene.DEFAULT_LAYER_NAME);

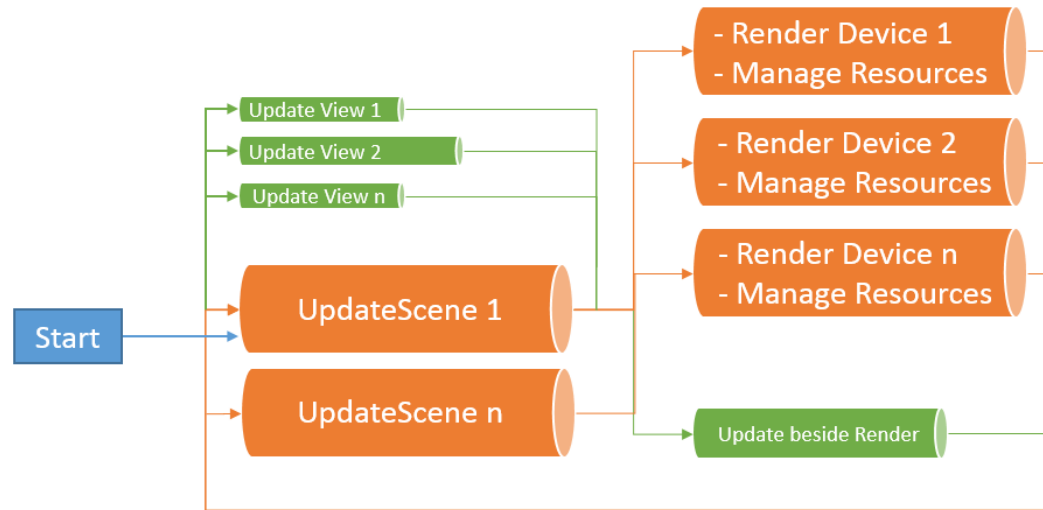
    // Create the wall in the middle
    AppendWallObjectToScene(manipulator);

    // Now create all pallets
    AppendPalletObjectsToScene(manipulator);
});

// Configure camera
camera.Position = new Vector3(30f, 30f, 30f);
camera.Target = new Vector3(0f, 0f, 0f);
camera.UpdateCamera();
```



### ■ 3D-Engine Hauptschleife per async-await



#### Update View 1

- DeviceChange
- SceneChange
- PrepareRendering
- Present
- SyncWithUI

#### UpdateScene (n)

- UpdateAnimations
- UpdateTransformations
- ManipulateScene
- Register/DeregisterView
- UpdateForView

#### - Render Device (n) - Manage Resources

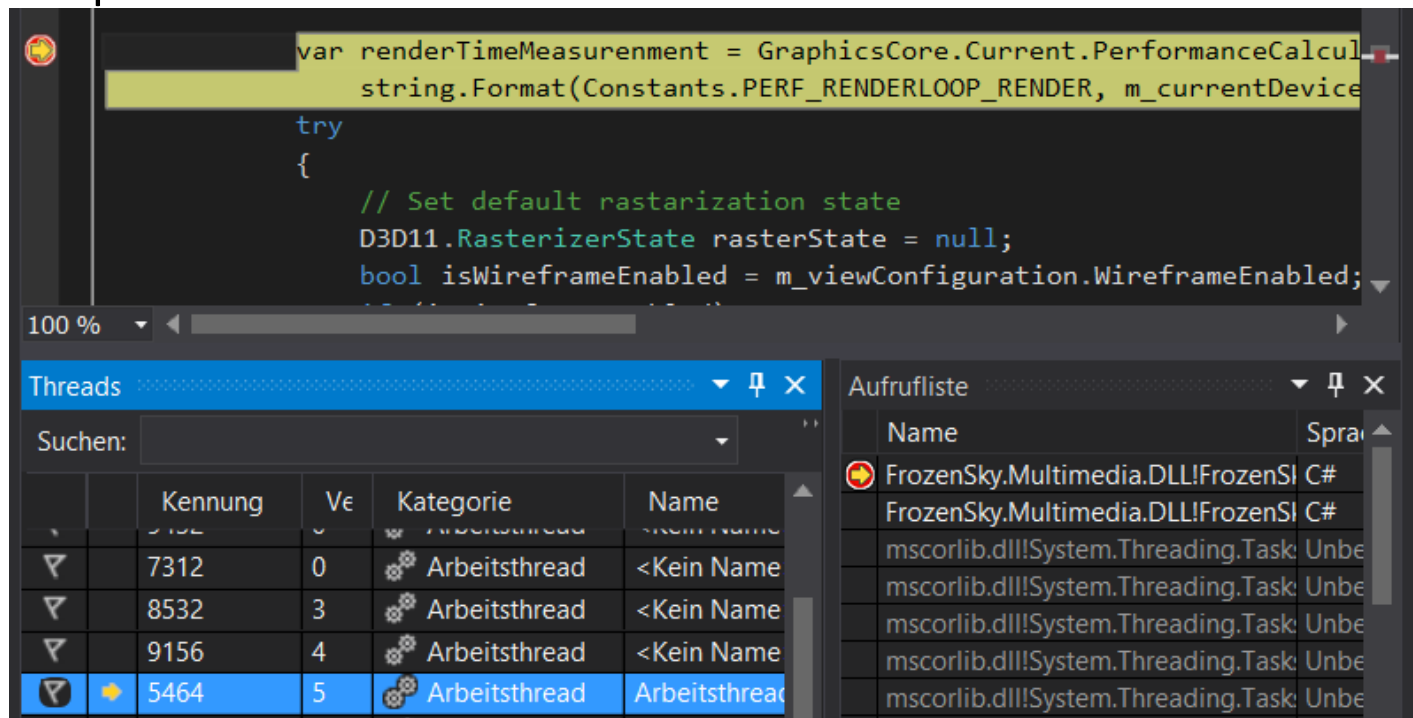
- LoadResources
- UnloadResources
- ConfigureShaders
- Render

#### Update beside Render

- VisibilityCheck
  - Clipping
  - LayerChecking
  - DetailChecking
- HitTesting



- Vor allem bei async-await prüfen, ob Methoden auch in den richtigen Threads aufgerufen werden!
- Beispiel: Hier sind wir in einem ThreadPool-Thread



The screenshot displays a Visual Studio IDE with a C# code editor at the top and two debugging windows below. The code editor shows a method with a try block containing several lines of code, including a comment about setting the default rasterization state. The 'Threads' window shows a list of threads with columns for 'Kennung', 'Ve', 'Kategorie', and 'Name'. The thread with ID 5464 is selected. The 'Aufrufliste' window shows a call stack with entries for 'FrozenSky.Multimedia.DLL!FrozenSI C#' and 'mscorlib.dll!System.Threading.Task Unbe'.

```
var renderTimeMeasurement = GraphicsCore.Current.PerformanceCalcul
string.Format(Constants.PERF_RENDERLOOP_RENDER, m_currentDevice
try
{
    // Set default rasterization state
    D3D11.RasterizerState rasterState = null;
    bool isWireframeEnabled = m_viewConfiguration.WireframeEnabled;
```

Kennung	Ve	Kategorie	Name
7312	0	Arbeitsthread	<Kein Name
8532	3	Arbeitsthread	<Kein Name
9156	4	Arbeitsthread	<Kein Name
5464	5	Arbeitsthread	Arbeitsthread

Name	Sprache
FrozenSky.Multimedia.DLL!FrozenSI	C#
FrozenSky.Multimedia.DLL!FrozenSI	C#
mscorlib.dll!System.Threading.Task	Unbe
mscorlib.dll!System.Threading.Task	Unbe
mscorlib.dll!System.Threading.Task	Unbe
mscorlib.dll!System.Threading.Task	Unbe
mscorlib.dll!System.Threading.Task	Unbe



- Nach await Prüfen, ob Context (noch) gültig!
  - Beispiel: Hier prüfe ich, ob das Control noch einen Parent hat

```
if(modelObject != null)
{
    while(this.Parent != null)
    {
        modelObject.CircleFieldPosition = new Vector2(
            modelObject.CircleFieldPosition.X > (float)Constants.CIRCLE_FIELD_WIDTH ? 0f : (flo
            modelObject.CircleFieldPosition.Y);
        TxtXPos.Text = modelObject.CircleFieldPosition.X.ToString("N0");

        await Task.Delay(10);
    }
}
```





- ThreadPool verteilt prinzipiell gut
- Async-Await gutes Werkzeug, um bestimmte Logikblöcke auf bestimmte Threads auszuführen
- Nachvollziehen von Fehlern z. T. schwieriger (Race-Condition)
- Teilweise neue Fehlerursachen durch Async-Await  
(wer prüft auch immer, ob es das Control noch gibt?)
- Vorgaben, welcher Thread wann was machen darf, helfen 😊  
(Chaos macht hier keinen Spaß)

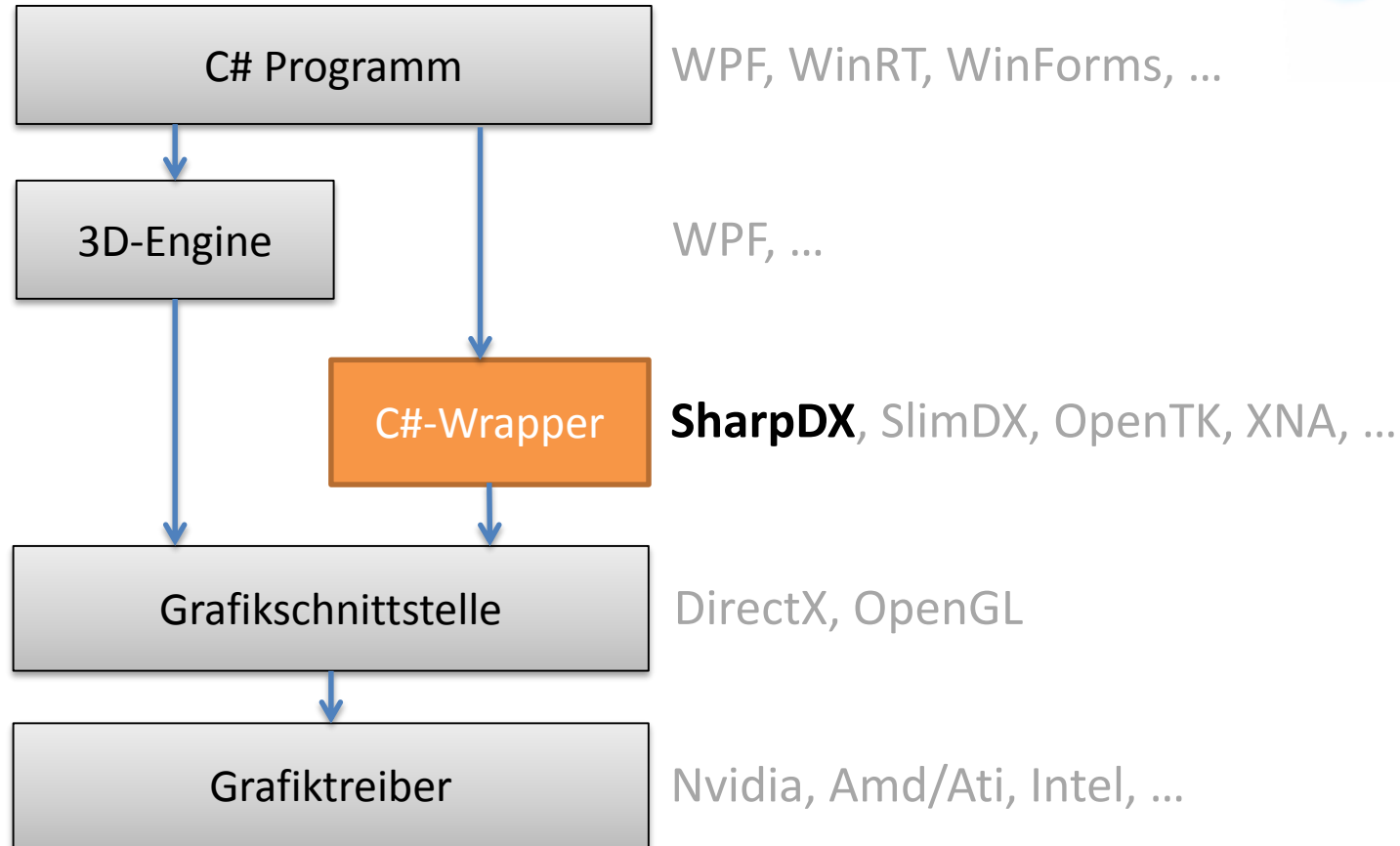


- Funktionsübersicht 3D-Engine
- Diverse Aspekte der Entwicklung
  - Verschiedene Plattformen
  - Multi-Core Entwicklung
  - **Native Schnittstellen**
  - Performance- und Speicher-Analyse
- Best practices...



# Native Schnittstellen

## Überblick





- Native Objekte nach Möglichkeit selber Disposed
  - Sehr nützliche Hilfsmethode

```
/// <summary>
/// Disposes the given object.
/// </summary>
internal static void SafeDispose<T>(ref T toDispose)
    where T : class, IDisposable
{
    toDispose = DisposeObject(toDispose);
}

/// <summary>
/// Disposes the given object and returns null.
/// </summary>
internal static T DisposeObject<T>(T objectToDispose)
    where T : class, IDisposable
{
    if (objectToDispose == null) { return null; }

    try { objectToDispose.Dispose(); }
    catch (Exception)
    {
        // Generic exception handling..
    }
    return null;
}
```

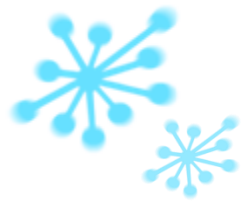


- Was gibt es da alles..?
  - X86
  - X64
  - ARM
  - Phone/X86
  - Phone/ARM
  - ...





- Genau informieren, was Threadsicher ist und was nicht!
  - Beispiel: Garbage Collector ruft Dispose aus anderen Thread auf
  - Oder: Dispose aus Gui- oder Worker-Thread aufrufen?
- Besser selbst um Dispose kümmern  
(GC sorgt häufig für böse Überraschungen)
- Speicher im Auge behalten (=> TaskManager)  
(Auch kleine Speicherlecks sind nach ein paar Tagen spürbar)



- Funktionsübersicht 3D-Engine
- Diverse Aspekte der Entwicklung
  - Verschiedene Plattformen
  - Multi-Core Entwicklung
  - Native Schnittstellen
  - **Performance- und Speicher-Analyse**
- Best practices...





# Performance- und Speicher-Analyse

## Visual Studio Profiler

**FrozenSky.Multimedia.Core.ViewRelatedSceneLayerSubset.UnsubscribeForPass**  
FrozenSky.Multimedia.dll

Aufrufende Funktionen → Aktuelle Funktion → Aufgerufene Funktionen

Aufrufende Funktionen	Aktuelle Funktion	Aufgerufene Funktionen
Unsubscribe 19.5%	UnsubscribeForPass 19.5% Funktionsrumpf < 0,1%	RemoveAt 19.4% [mscorlib.ni.dll] < 0,1%

Verknüpfte Ansichten: [Aufrufer/Aufgerufener](#) [Funktionen](#) Leistungsmetrik: **Inklusive Samplings in %**

**Funktionscodeansicht**  
d:\Projects\TfsRkoenigCode\FrozenSky\FrozenSky.Multimedia\Core\_Scene\ViewRelatedSceneLayerSubset.cs

```
/// <param name="passInfo">The pass to unsubscribe from.</param>
/// <param name="sceneObject">The scene object to unsubscribe.</param>
internal void UnsubscribeForPass(RenderPassInfo passInfo, SceneObject sceneObject)
{
    List<RenderPassSubscription> subscriptionList = m_objectsPassCollections[passInfo];
    for (int loop = 0; loop < subscriptionList.Count; loop++)
    {
        if (subscriptionList[loop].SceneObject == sceneObject)
        {
            subscriptionList.RemoveAt(loop);
            return;
        }
    }
}

/// <summary>
/// Handles changed object visibility.
```

100 %

# Performance- und Speicher-Analyse

## PerfView

Stacks(23.809.780 metric) FrozenSky.Samples.WpfSampleContainer.1.gcdump in PerfView (C:\Program Files (x86)\PerfView\FrozenSky... —

File Diff Help Stack View Help (F1) Understanding Perf Data Starting an Analysis Troubleshooting Tips

Update Back Forward Totals Metric: 23.809.780,0 Count: 433.062,3 UnreachableMemory: 6,503MB (21,5%) Heap Sampled: Mean Count Multiplier 1,95 Mean Size Multiplier 1,47

Start: 0 End: 0,000 Priority: v4.0.30319\%!->-1;v2.0.507. Pri1Only: Find:

GroupPats: [group Framework] mscorlib! => LIB;S Fold%: 0 FoldPats: [];mscorlib!String IncPats: ExcPats: [not reachable from roots]

By Name RefFrom-RefTo RefTree Referred-From Refs-To Notes

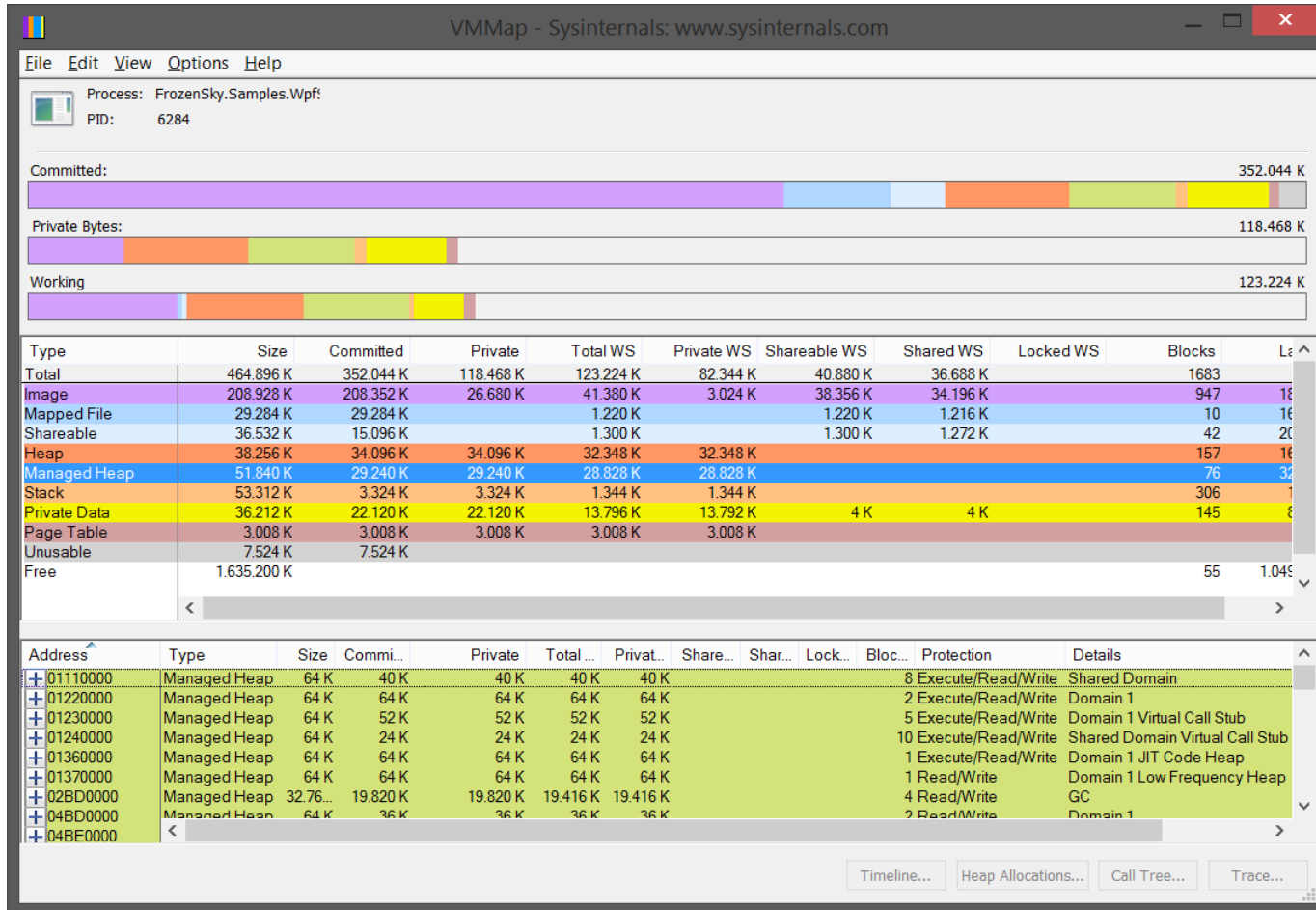
Name	Exc %	Exc	Exc Ct	Inc %	Inc	Inc Ct	Fold	Fold Ct
LIB <<microsoftlib!List<FrozenSky.Multimedia.Objects.Vertex>>>	20.4	4,863,312	40	20.4	4,863,312.0	40	4,862,832	20
FrozenSky.Multimedia!FrozenSky.Multimedia.Objects.GenericObject	8.6	2,048,704	8,003	52.9	12,600,560.0	337,932	0	0
FrozenSky.Multimedia!FrozenSky.Multimedia.Drawing3D.TypeSafeConstantBufferResource <FrozenSky.Multimed	5.5	1,312,805	8,005	6.6	1,568,047.0	15,981	0	0
LIB <<microsoftlib!Dictionary<List<FrozenSky.Multimedia.Core.RenderPassSubscription>,Int32>>>	4.4	1,056,850	24,031	4.4	1,056,850.0	24,031	192,455	8,019
LIB <<microsoftlib!Dictionary<FrozenSky.Multimedia.Core.VisibilityCheckData,Int32>>>	4.4	1,056,815	24,028	4.4	1,056,815.0	24,028	192,405	8,016
LIB <<microsoftlib!Dictionary<FrozenSky.Multimedia.Drawing3D.ObjectRenderParameters,Int32>>>	4.4	1,056,677	24,025	4.4	1,056,677.0	24,025	192,405	8,016
LIB <<microsoftlib!Dictionary<FrozenSky.Multimedia.Drawing3D.GeometryResource,Int32>>>	4.4	1,056,677	24,025	4.4	1,056,677.0	24,025	192,405	8,016
LIB <<microsoftlib!List<List<FrozenSky.Multimedia.Core.RenderPassSubscription>>>>	2.7	640,669	16,016	2.7	640,668.8	16,016	222	4
LIB <<microsoftlib!List<FrozenSky.Multimedia.Core.VisibilityCheckFilterStageData>>>	2.7	640,405	16,010	2.7	640,404.9	16,010	448,283	8,005
LIB <<microsoftlib!List<FrozenSky.Multimedia.Core.VisibilityCheckData>>>	2.7	640,402	16,008	8.5	2,033,060.0	62,752	448,300	8,005
LIB <<microsoftlib!List<FrozenSky.Multimedia.Drawing3D.GeometryResource>>>	2.7	640,258	16,004	2.7	640,258.1	16,004	448,162	8,003
LIB <<microsoftlib!List<FrozenSky.Multimedia.Drawing3D.ObjectRenderParameters>>>	2.7	640,258	16,004	3.2	759,690.9	17,761	448,162	8,003
LIB <<microsoftlib!Object>>	2.4	570,346	47,526	2.4	570,345.6	47,526	0	0
LIB <<microsoftlib!Dictionary<FrozenSky.Util.NamedOrGenericKey,FrozenSky.Multimedia.Core.ResourceDictionary+	2.4	565,424	34	34.2	8,154,099.0	41,597	2,612	23
FrozenSky.Multimedia!FrozenSky.Multimedia.Drawing3D.ObjectRenderParameters	2.3	544,204	8,003	2.3	548,247.5	8,028	0	0
[Pinned handle]	1.8	439,750	5,655	2.2	515,675.1	8,252	439,750	5,654
LIB <<microsoftlib!List<FrozenSky.Multimedia.Core.IAnimation>>>	1.6	384,273	16,011	1.6	384,272.5	16,011	0	0
LIB <<microsoftlib!Dictionary<FrozenSky.Multimedia.Core.VisibilityCheckFilterStageData,Int32>>>	1.6	384,243	8,005	1.6	384,242.8	8,005	0	0

Notes typed here will be saved when the view is saved. F2 will hide/unhide.

Completed: Computing Stack Traces (Elapsed Time: 0,313 sec) Ready Log Cancel

# Performance- und Speicher-Analyse

## VMMMap



# Performance- und Speicher-Analyse

## Grafik-Debugging



The screenshot displays the Microsoft Visual Studio interface for graphics debugging. The main window shows a 3D scene with a green cube on a tiled floor. The interface is divided into several panels:

- Graphics Pixel History:** Shows the final frame buffer with a color of 0.007843137, 0.345098039, 1.000000000. It also displays the render alpha value and the number of pixels (385, 197).
- Graphics Pipeline Stages:** Shows the pipeline stages: Input Assembler, Vertex Shader, Pixel Shader, and Output Merger. The Vertex Shader stage is currently selected.
- Graphics Event List:** A list of graphics events, including DrawIndexed(612,0,0) and DrawIndexed(36,0,0). The selected event is DrawIndexed(612,0,0).
- Properties:** Shows the Direct3D Information for the selected event, including the 10-bit XR High Color Format and DirectCompute CS-4.x.

The bottom status bar indicates the application is ready.



# Performance- und Speicher-Analyse

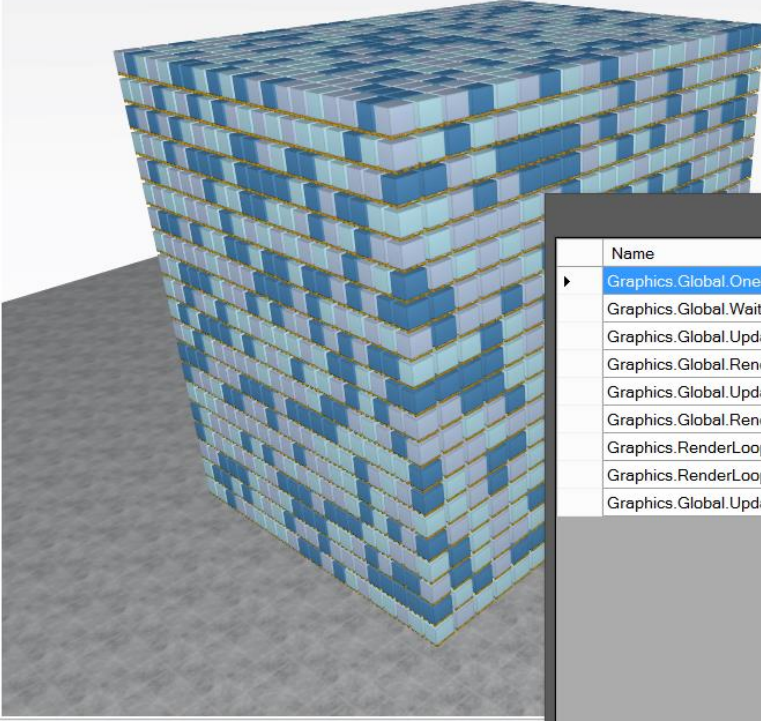
Eigene Auswertungen



FrozenSky Win.Forms Samples

NVIDIA Quadro K2000M

Single Model Pallets Transparent Pallets Async Animation



Performance

Name	Milliseconds (avg)
▶ Graphics.Global.OneFrame	24
Graphics.Global.WaitTime	7
Graphics.Global.UpdateAndPrepare	3
Graphics.Global.RenderAndUpdateBeside	14
Graphics.Global.UpdateBeside (Scene: 0)	1
Graphics.Global.Render (Device: NVIDIA Quadro K2000M)	13
Graphics.RenderLoop.Render (Scene: 0, View: 1)	13
Graphics.RenderLoop.Present (Scene: 0, View: 1)	0
Graphics.Global.UpdateBeside (Scene: 1)	0





- **Sehr viele Tools, manche gut, manche weniger**  
(zumindest nicht im gezeigten Beispiel)
- **Visual Studio hierbei viel mächtiger als früher**  
(aber z. T. auch etwas sperrig)
- **Wichtig ist es für Speicher und Performance jeweils ein gutes Analyse-Tool zu kennen**



- Funktionsübersicht 3D-Engine
- Diverse Aspekte der Entwicklung
  - Verschiedene Plattformen
  - Multi-Core Entwicklung
  - Native Schnittstellen
  - Performance- und Speicher-Analyse
- **Best practices...**





### ■ Internet-Quelle im Code angeben

#### ■ Beispiel: Wer weiß, was die „1“ macht?

```
// Present all rendered stuff on screen
// First parameter indicates synchronization with vertical blank
// see http://msdn.microsoft.com/en-us/library/windows/desktop/bb174576\(v=vs.85\).aspx
// see example http://msdn.microsoft.com/en-us/library/windows/apps/hh825871.aspx
```

```
m_swapChain.Present(1, DXGI.PresentFlags.None);
```

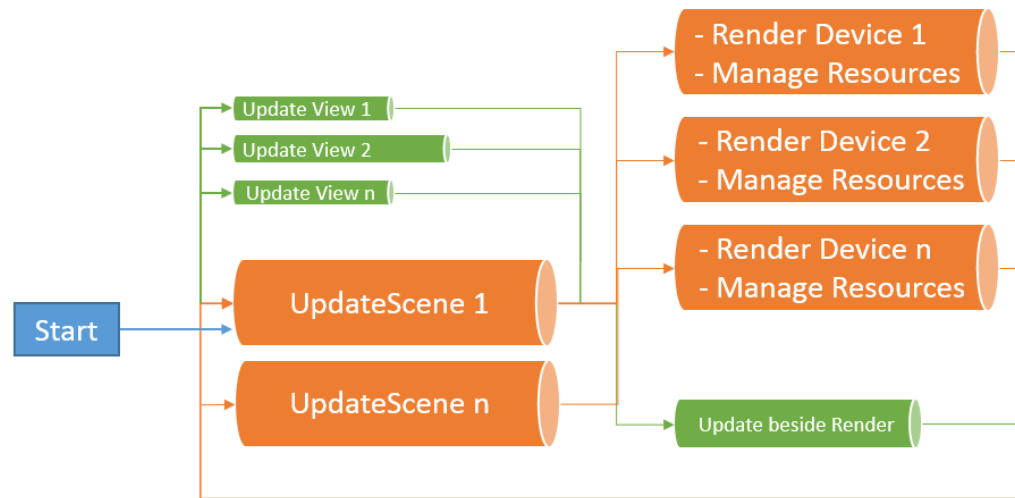
#### ■ Beispiel: Was tun bei Bugs im Triangulator?

```
/// <summary>
/// A cutting ears triangulator for simple polygons with no holes.  $O(n^2)$ 
/// This algorithm is based on the implementation of the Helix Toolkit (http://helixtoolkit.codeplex.com/)
/// </summary>
/// <remarks>
/// Regarding the Helix Toolkit:
/// Based on http://www.flipcode.com/archives/Efficient Polygon Triangulation.shtml
/// References
/// http://en.wikipedia.org/wiki/Polygon triangulation
/// http://computation.cs.cinvestav.mx/~anzures/geom/triangulation.php
/// http://www.codeproject.com/KB/recipes/cspolygontriangulation.aspx
/// </remarks>
public static class CuttingEarsTriangulator
{
```





- Bildchen zeichnen... ;)
  - Beispiel: Übersicht welcher Thread zu welcher Zeit was macht



### Update View 1

- DeviceChange
- SceneChange
- PrepareRendering
- Present
- SyncWithUI

### UpdateScene (n)

- UpdateAnimations
- UpdateTransformations
- ManipulateScene
- Register/DeregisterView
- UpdateForView

### - Render Device (n) - Manage Resources

- LoadResources
- UnloadResources
- ConfigureShaders
- Render

### Update beside Render

- VisibilityCheck
  - Clipping
  - LayerChecking
  - DetailChecking
- HitTesting





- Aufwändige Optimierungen erst bei Bedarf
  - Beispiel: Sortieren transparenter Objekte pro Frame  
(eigentlich völliger Käse, reicht aber so...)

```
// TODO: Trigger some other logic to update transparent object order
// TODO: Performance improvement!!!
ICamera3D camera = m_viewInformation.Camera;
m_objectsPassTransparentRender.Sort(new Comparison<RenderPassSubscription>((left, right) =>
{
    SceneSpacialObject leftSpacial = left.SceneObject as SceneSpacialObject;
    SceneSpacialObject rightSpacial = right.SceneObject as SceneSpacialObject;
    if ((leftSpacial != null) && (rightSpacial != null))
    {
        float leftDistance = (camera.Position - leftSpacial.Position).LengthSquared();
        float rightDistance = (camera.Position - rightSpacial.Position).LengthSquared();
        return rightDistance.CompareTo(leftDistance);
    }
    else if (leftSpacial != null) { return -1; }
    else if (rightSpacial != null) { return 1; }
    {
        return 0;
    }
}));
```





Vielen Dank für eure  
Aufmerksamkeit!

