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

Über mich

Kontakt

E-Mail: roland.koenig@rolandk.de

Homepage: www.rolandk.de/wp

Berufliches

Schwerpunkt: Produktentwicklung

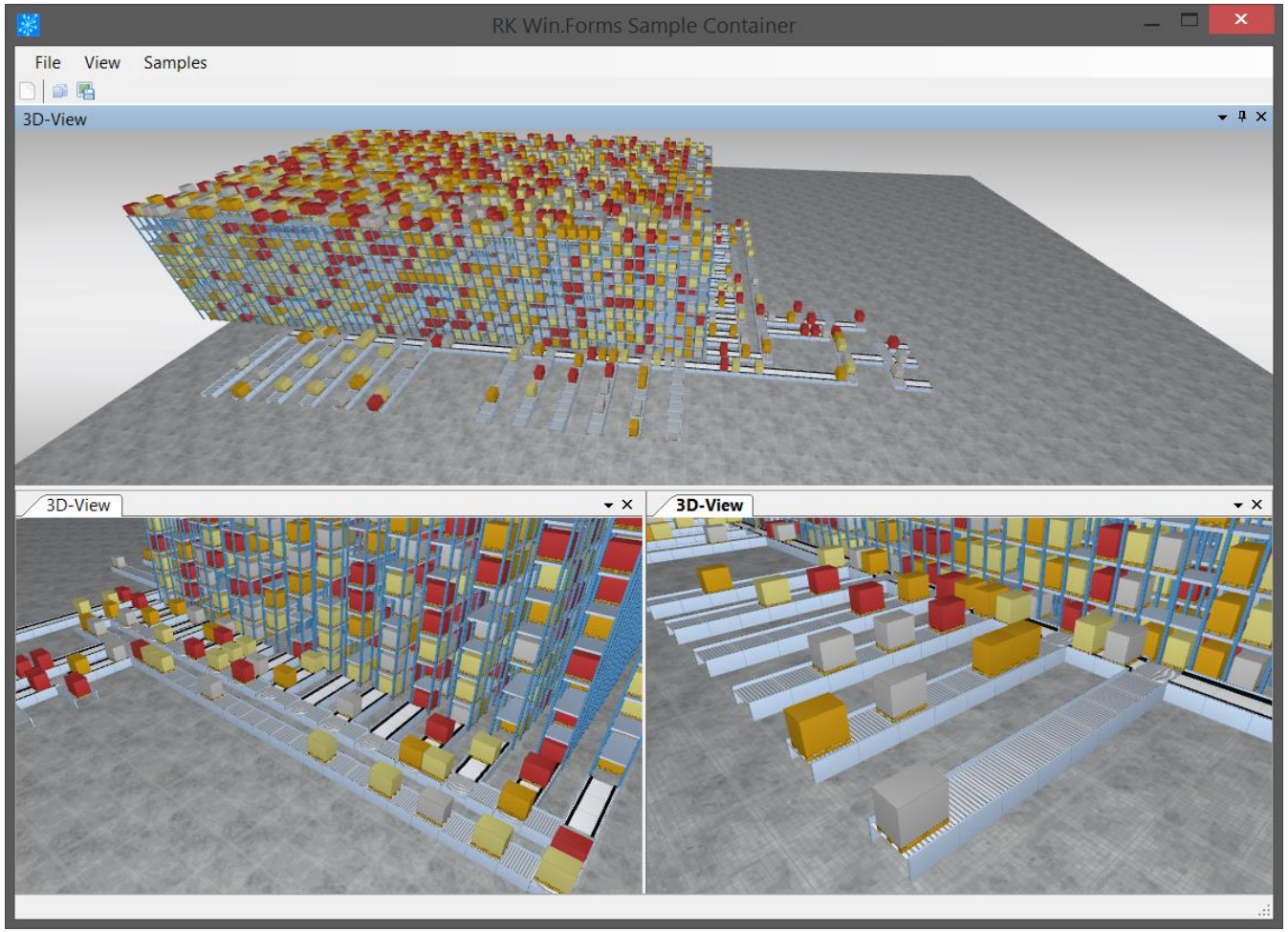
Arbeitgeber: IGZ Logistics + IT, Falkenberg

Agenda

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

Funktionsübersicht 3D-Engine

Beispiel



Funktionsübersicht 3D-Engine

Performance

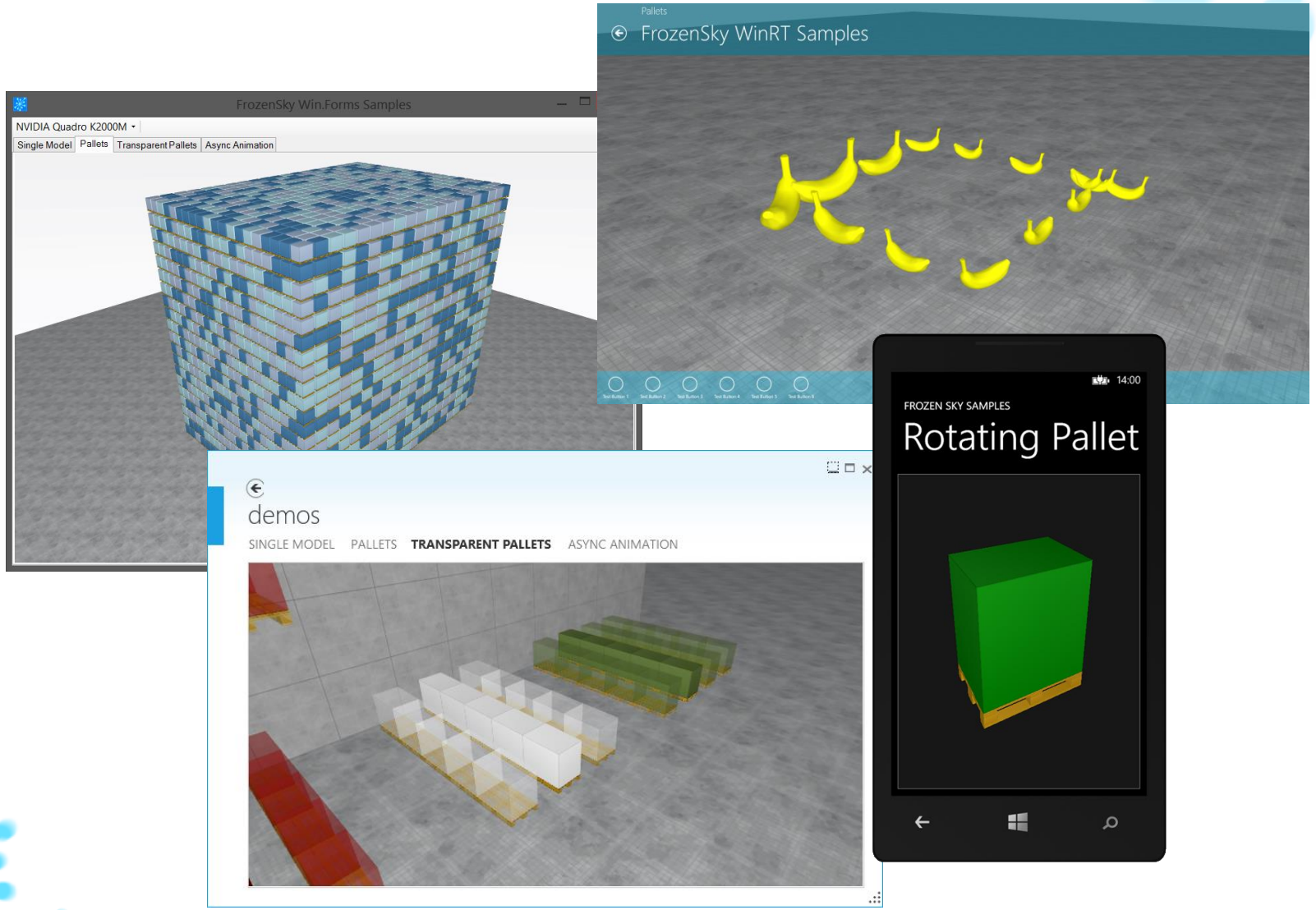
- Zeit pro Frame (Bild): ca. **33ms**

  (bei 30 FPS)

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

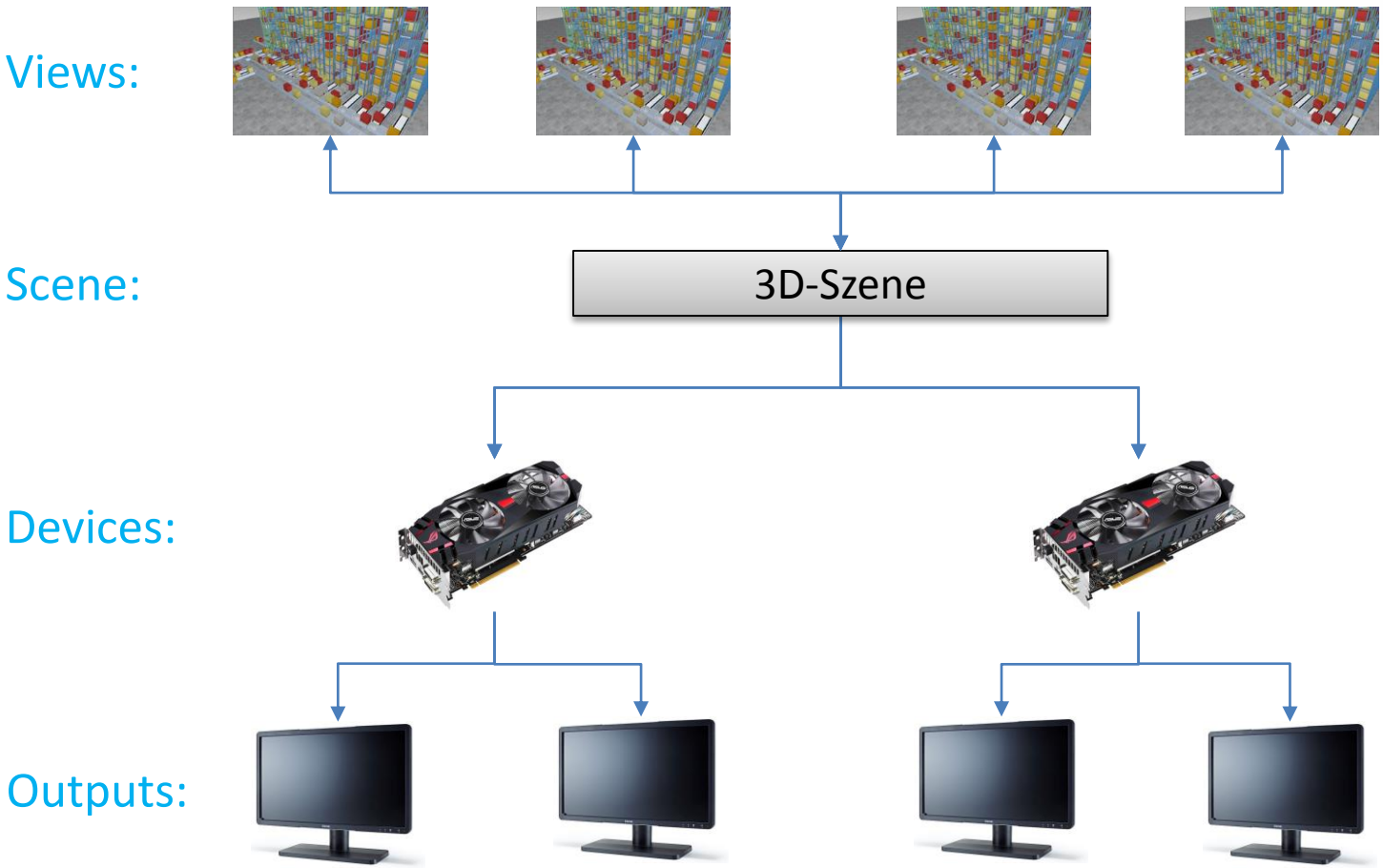
Funktionsübersicht 3D-Engine

Plattformen



Funktionsübersicht 3D-Engine

Hardware-Anbindung

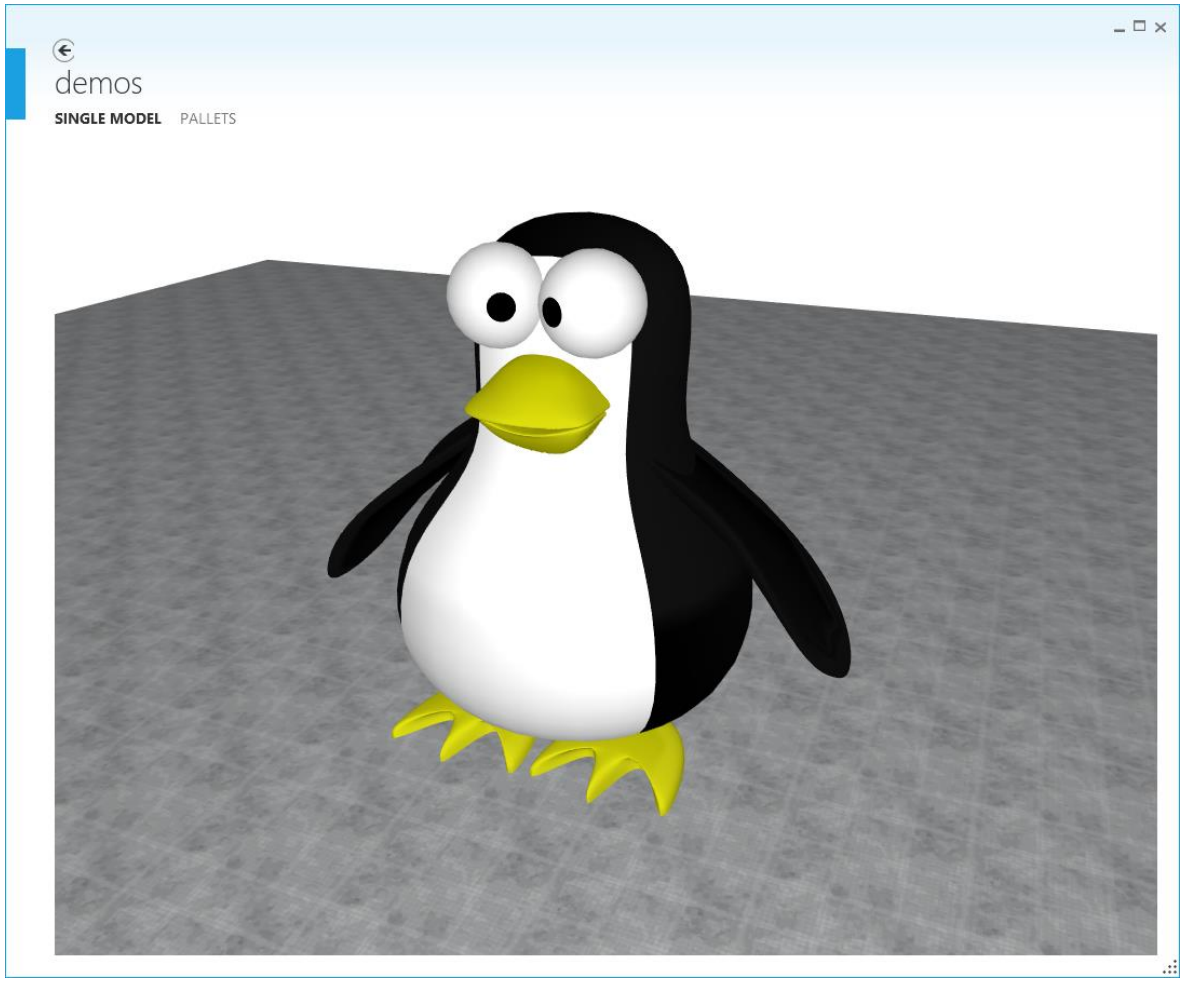


Agenda

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

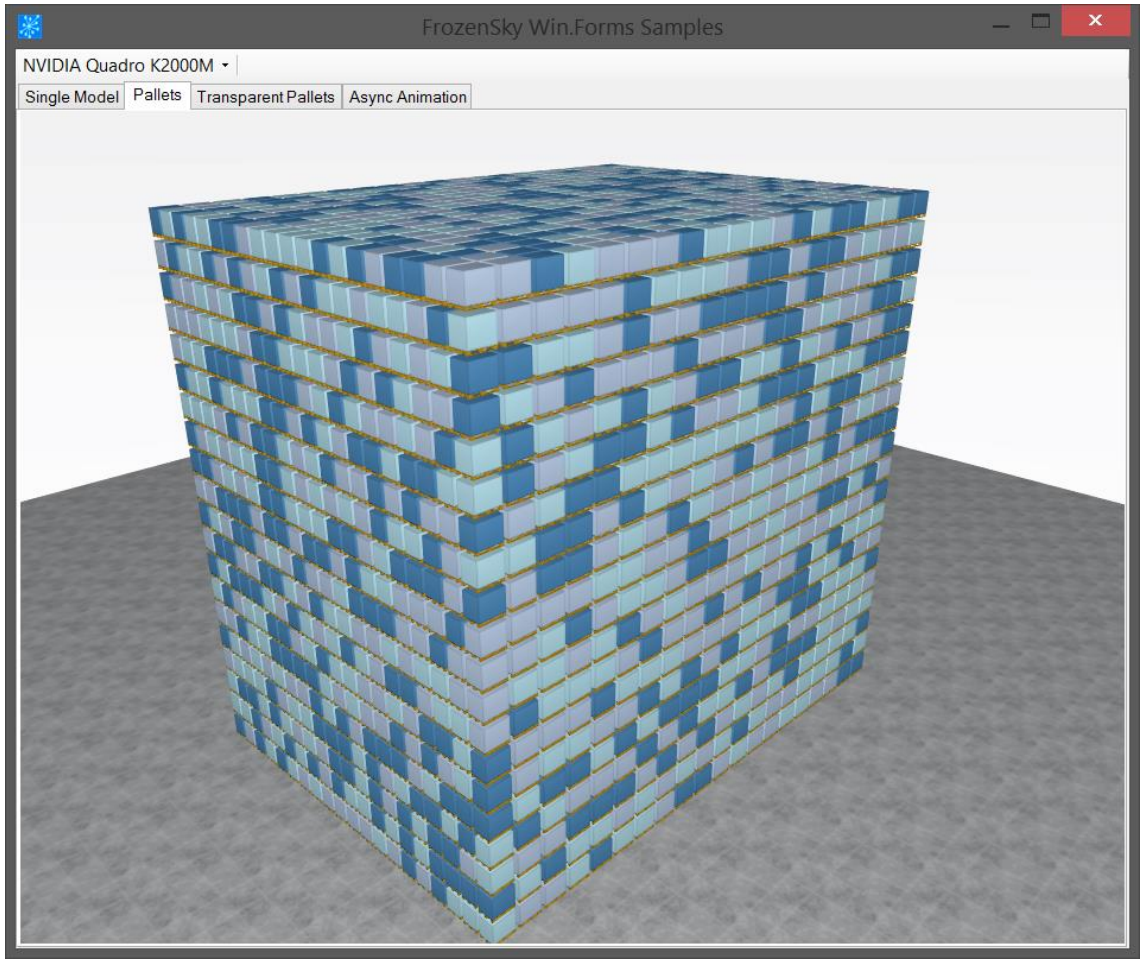
Verschiedene Plattformen

Beispiel WPF



Verschiedene Plattformen

Beispiel Win.Forms



Verschiedene Plattformen

Beispiel WinRT

Verschiedene Plattformen

Erfahrungen

- 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…

Agenda

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

Beispiel

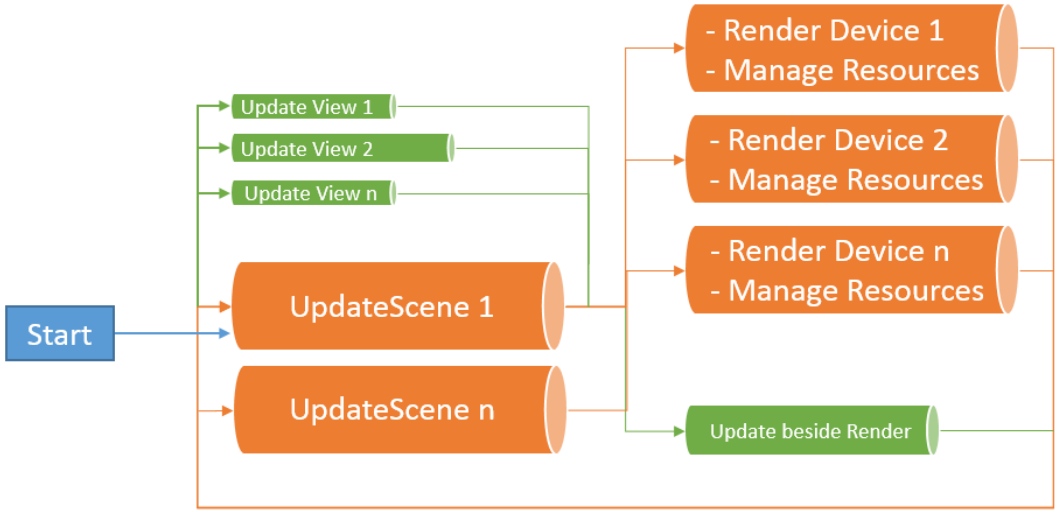
- Manipulation der Scene

```
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();
```

Multi-Core Entwicklung

Beispiel

- 3D-Engine Hauptschleife per async-await



Update View 1
- DeviceChange
- SceneChange
- PrepareRendering
- Present
- SyncWithUI

UpdateScene (n)
- UpdateAnimations
- UpdateTransformations
- ManipulateScene
- Register/DeregisterView
- UpdateForView

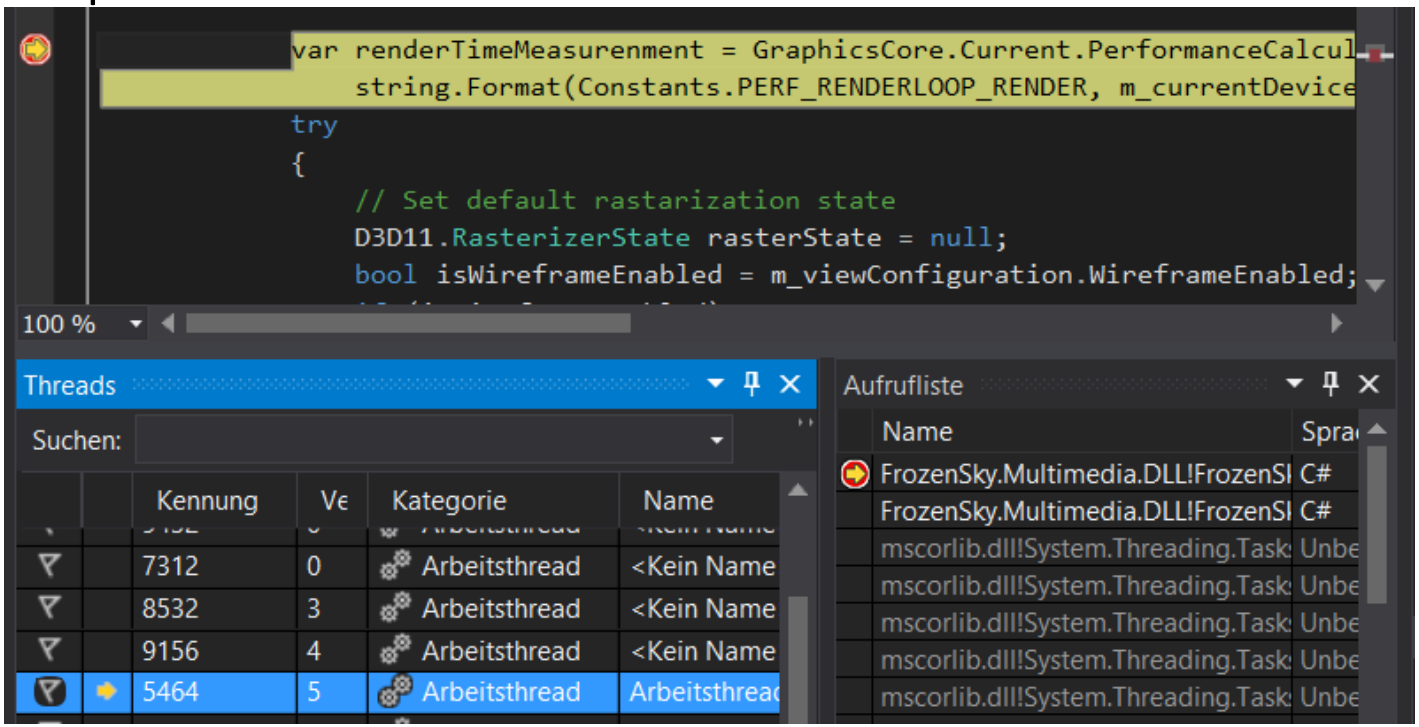
- Render Device (n)
- Manage Resources
- LoadResources
- UnloadResources
- ConfigureShaders
- Render

Update beside Render
- VisibilityCheck
  - Clipping
  - LayerChecking
  - DetailChecking
- HitTesting

Multi-Core Entwicklung

Probleme

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



Probleme

- 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);
    }
}
```

Multi-Core Entwicklung

Erfahrungen

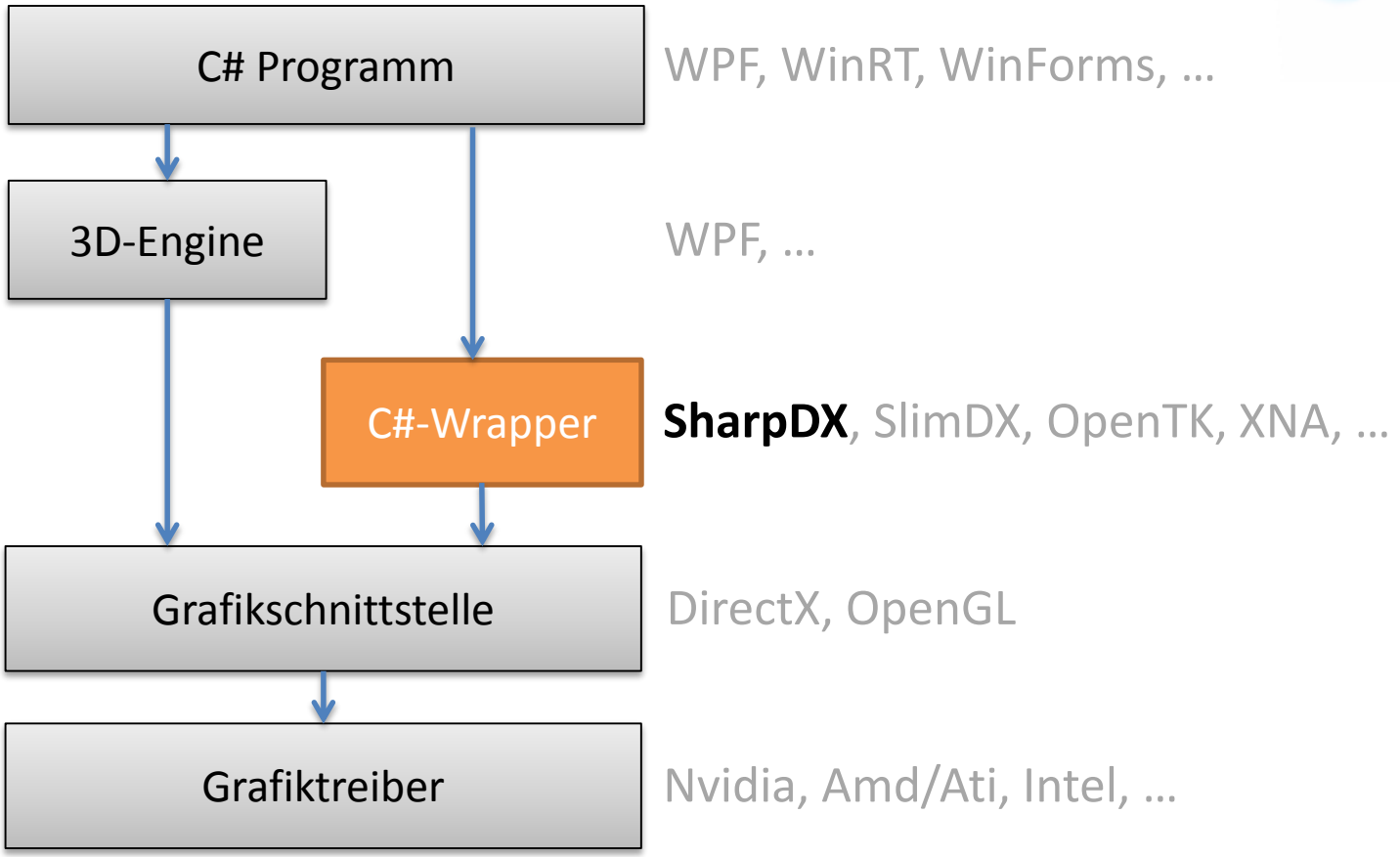
- ThreadPool verteilt prinzipiell gut
- Async-Await gutes Werkzeug, um bestimmte Logikblöcke auf bestimmte Threads auszuführen
- Nachvollziehen von Fehlern z. T. schwieriger (Race-Condition)
- Teilweise neue Fehlerursachen durch Async-Await
  (wer prüft auch immer, ob es das Control noch gibt?)
- Vorgaben,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

Agenda

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

Native Schnittstellen

Überblick



Native Schnittstellen

Speichermanagement

- Native Objekte nach Möglichkeit selber Disposen
  - 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;
}
```

Native Schnittstellen

Prozessorarchitektur

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

Native Schnittstellen

Erfahrungen

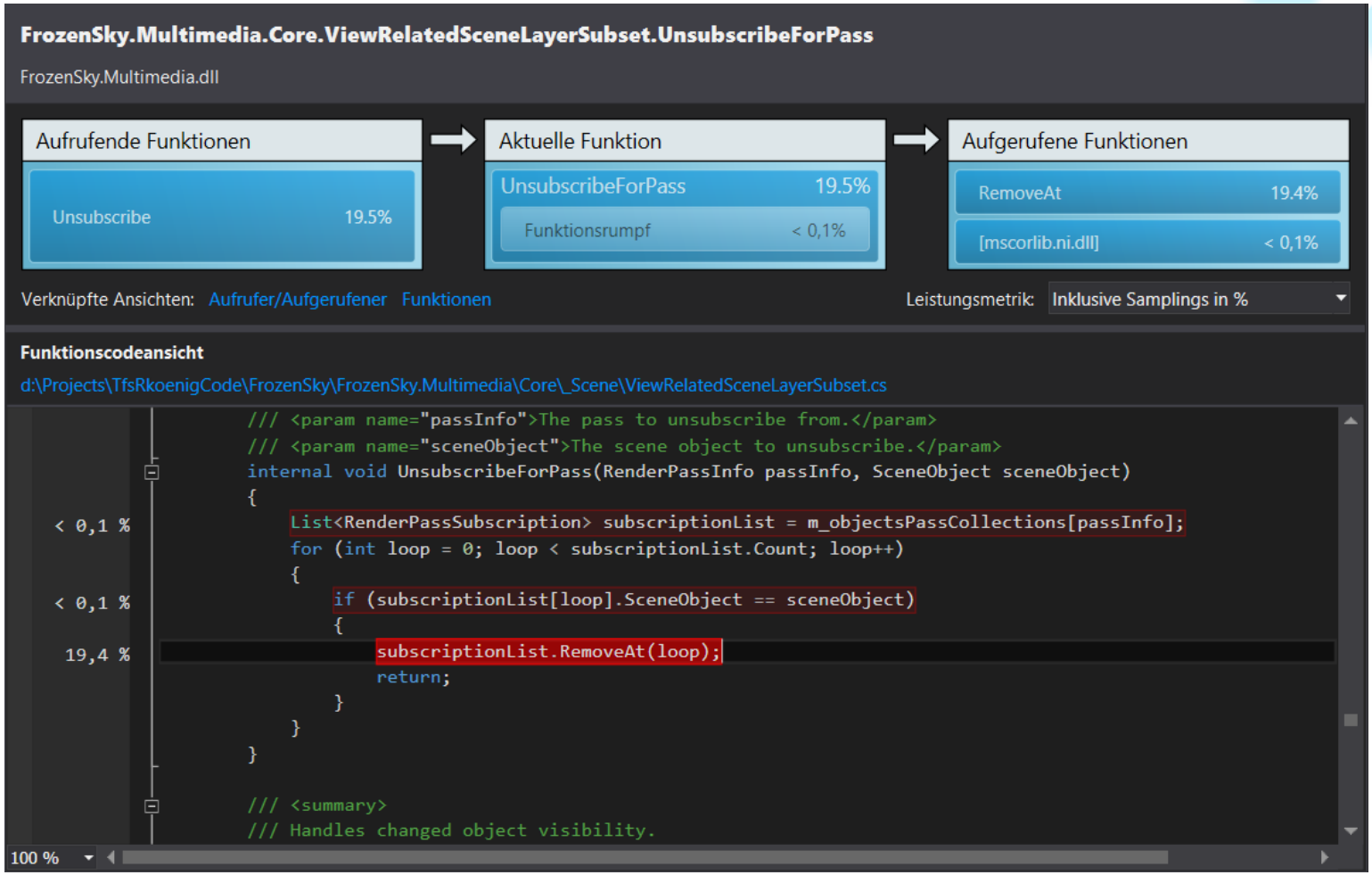
- Genau informieren, was Threadsicher ist und was nicht!
  - Beispiel: Garbage Collector ruft Dispose aus anderen Thread auf
  - Oder: Dispose aus Gui- oder Worker-Thread aufrufen?
- Besser selbst um Dispose kümmern
  (GC sorgt häufig für böse Überraschungen)
- Speicher im Auge behalten (=> TaskManager)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

Agenda

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

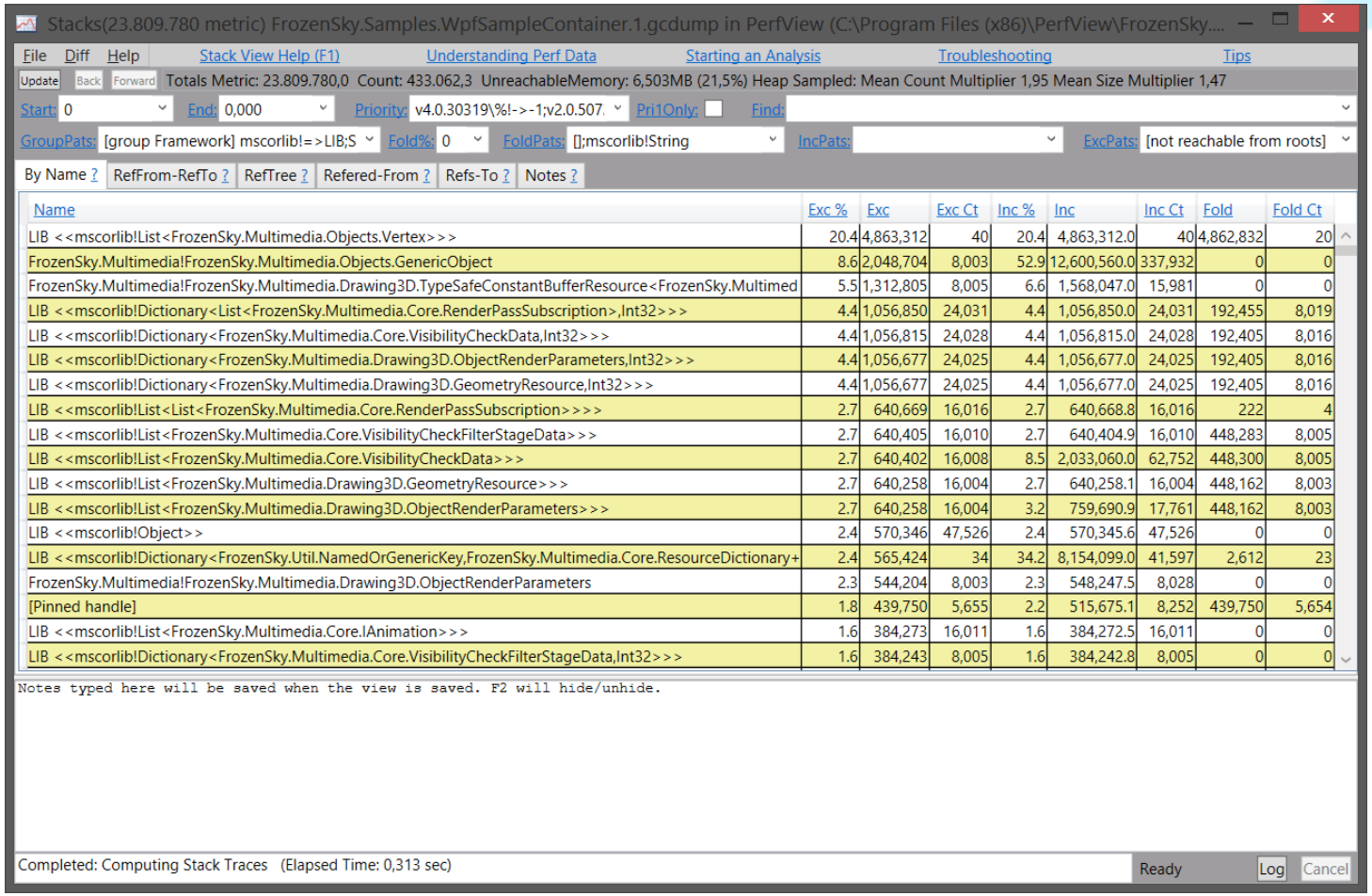
Performance- und Speicher-Analyse

Visual Studio Profiler



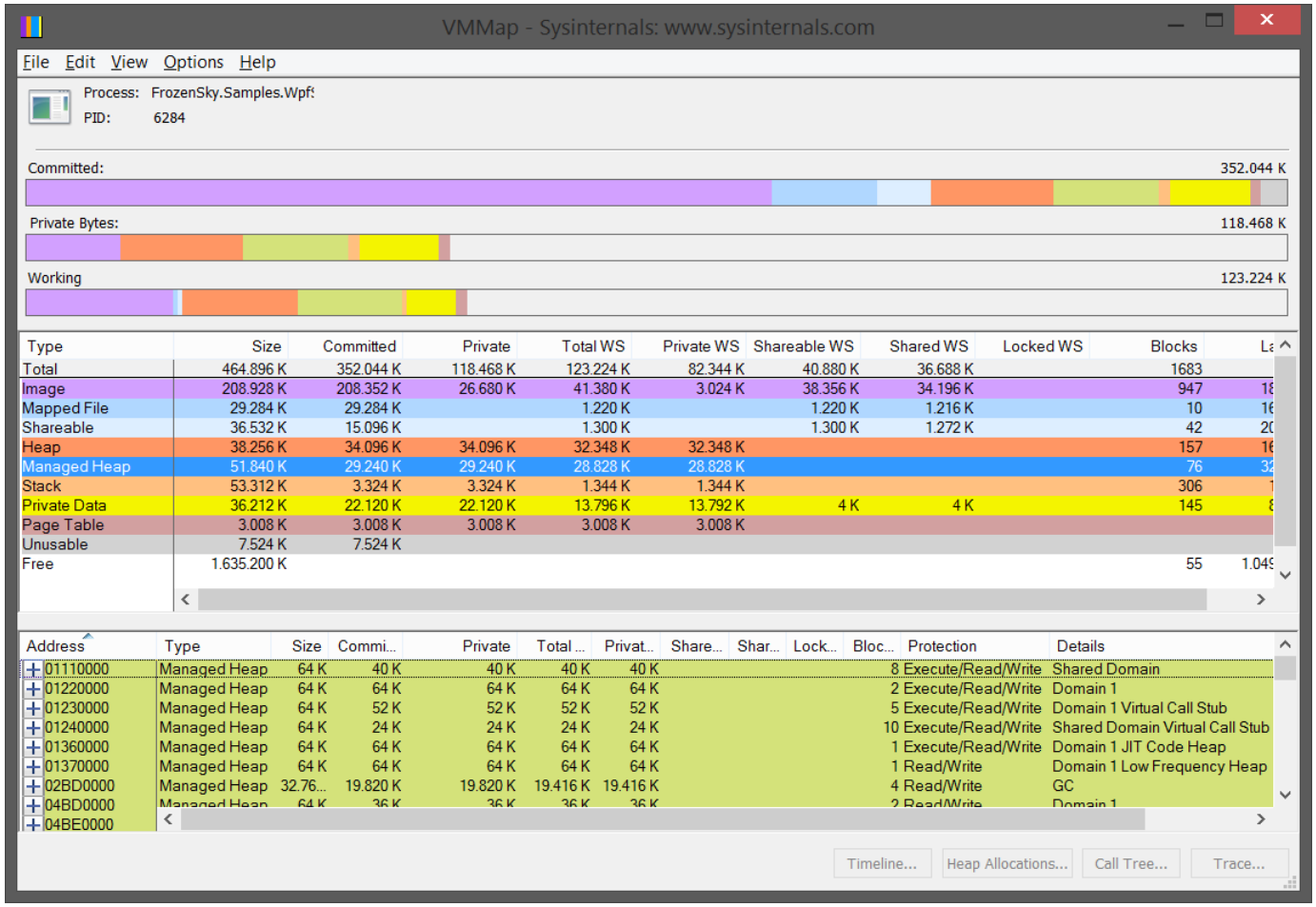
Performance- und Speicher-Analyse

PerfView



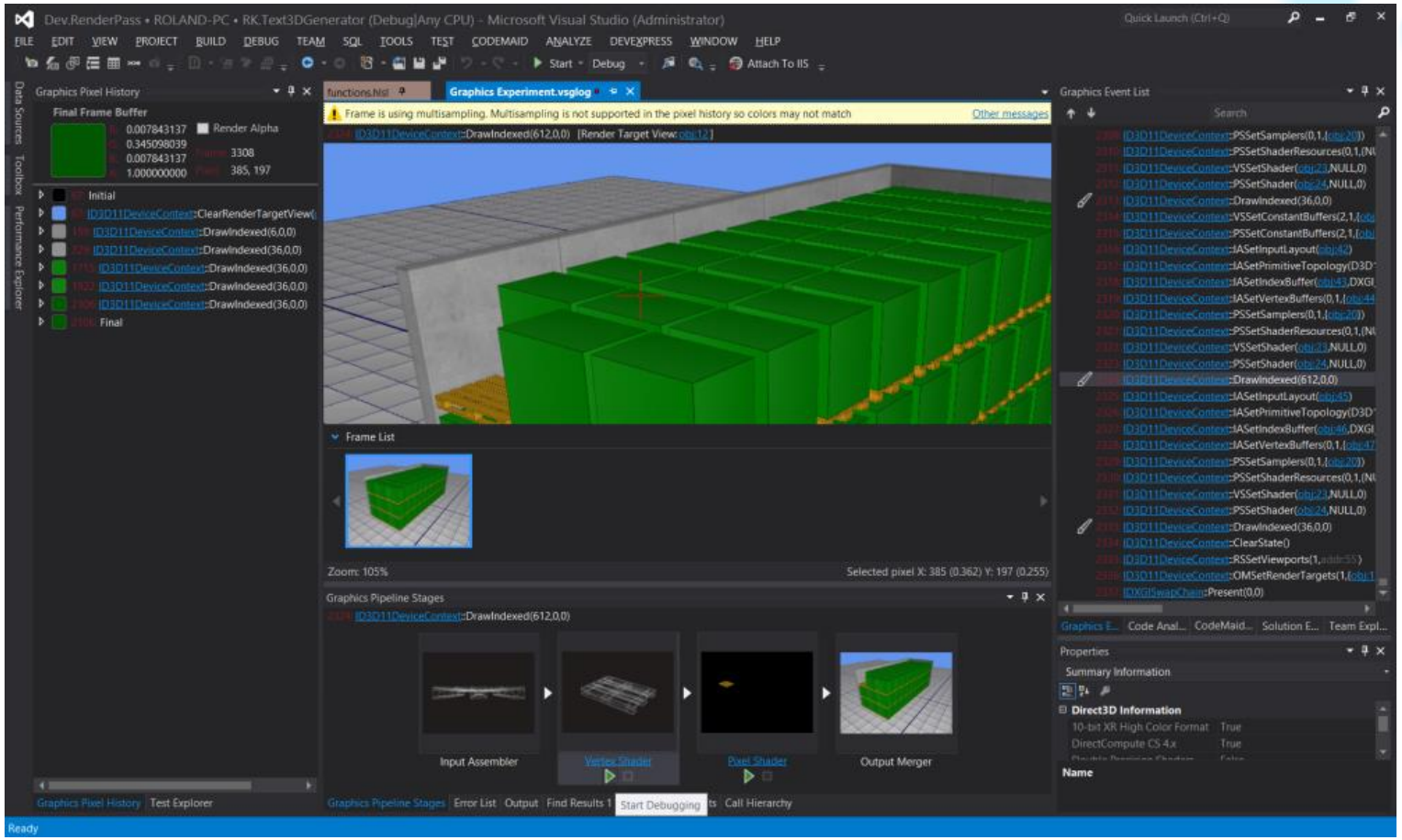
Performance- und Speicher-Analyse

VMMap



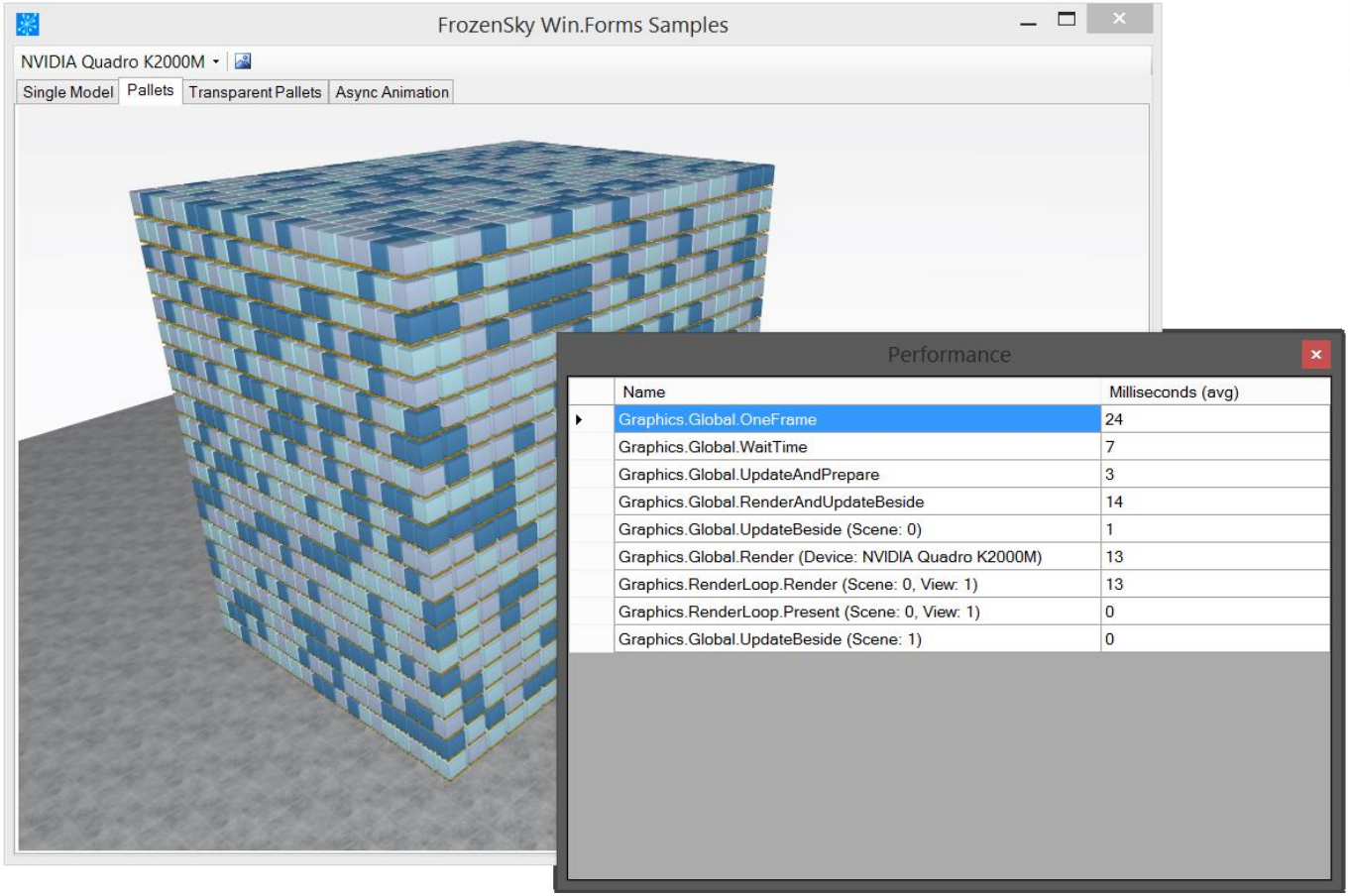
Performance- und Speicher-Analyse

Grafik-Debugging



Performance- und Speicher-Analyse

Eigene Auswertungen



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

Performance- und Speicher-Analyse

Erfahrungen

- 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

Agenda

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

Best practices

Quellenangaben

- 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);
```

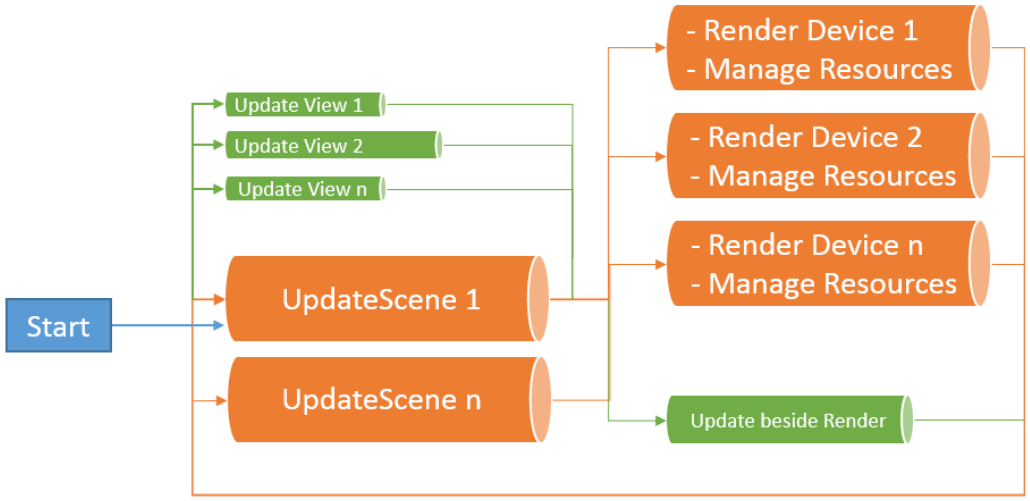
  - Beispiel: Was tun bei Bugs im Triangulator?

```
/// <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://computacion.cs.cinvestav.mx/~anzures/geom/triangulation.php
/// http://www.codeproject.com/KB/recipes/cspolygontriangulation.aspx
/// </remarks>
public static class CuttingEarsTriangulator
{
```

Best practices

Diagramme

- 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

Best practices

Optimierung

- 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!