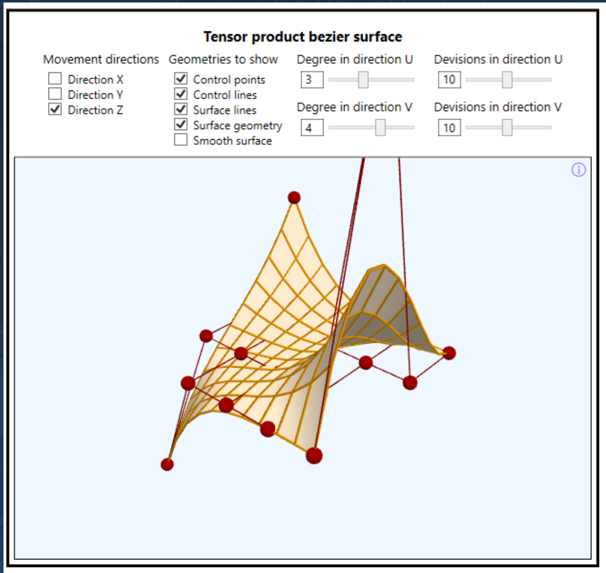
From XAML to cross-platform development with Unity


Tensor product bezier surface

Movement directions
- [ ] Direction X
- [ ] Direction Y
- [✓] Direction Z

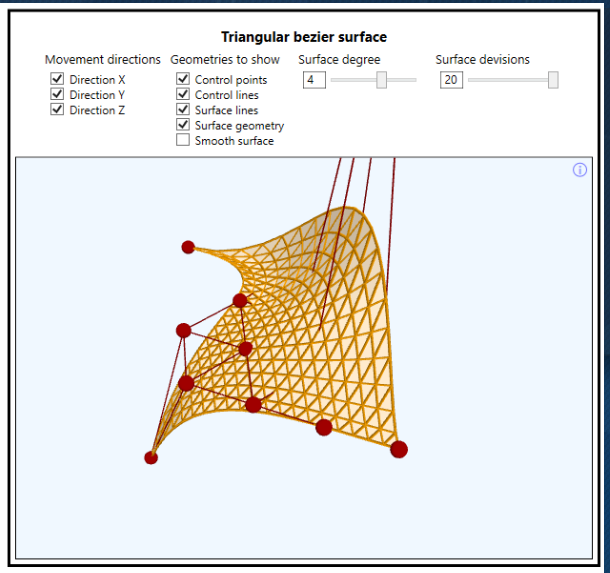
Geometries to show
- [✓] Control points
- [✓] Control lines
- [✓] Surface lines
- [✓] Surface geometry
- [ ] Smooth surface

Degree in direction U: 3

Degree in direction V: 4

Devisions in direction U: 10

Devisions in direction V: 10


Triangular bezier surface

Movement directions
- [✓] Direction X
- [✓] Direction Y
- [✓] Direction Z

Geometries to show
- [✓] Control points
- [✓] Control lines
- [✓] Surface lines
- [✓] Surface geometry
- [ ] Smooth surface

Surface degree: 4

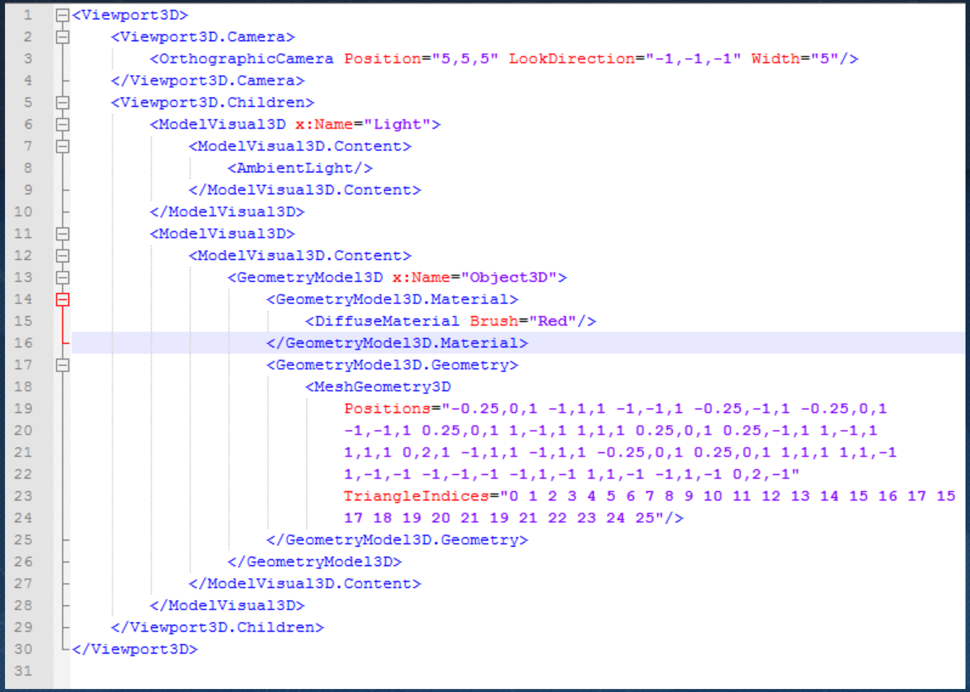
Surface devisions: 20

Progress®

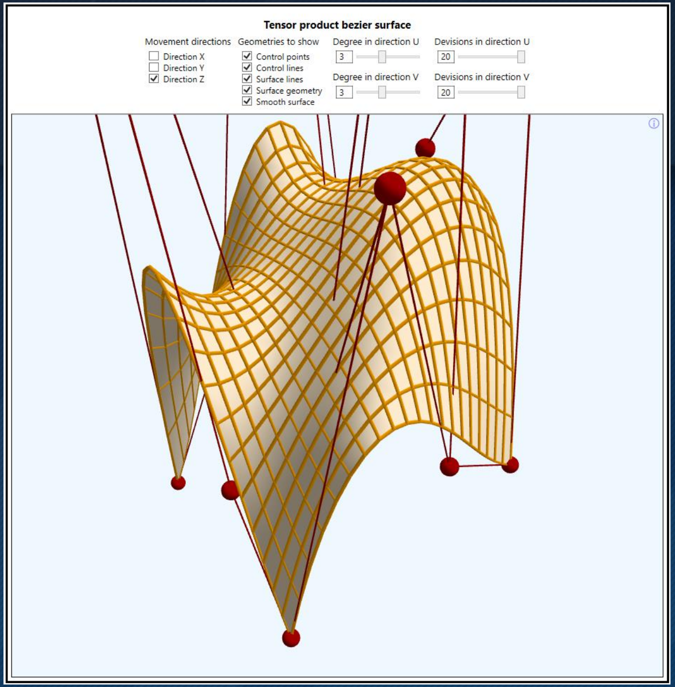
by Deyan Yosifov

Senior Software Developer @ Progress Telerik

What is WPF 3D?



```xml
<Viewport3D>
    <Viewport3D.Camera>
        <OrthographicCamera Position="5,5,5" LookDirection="-1,-1,-1" Width="5"/>
    </Viewport3D.Camera>
    <Viewport3D.Children>
        <ModelVisual3D x:Name="Light">
            <ModelVisual3D.Content>
                <AmbientLight/>
            </ModelVisual3D.Content>
        </ModelVisual3D>
        <ModelVisual3D>
            <ModelVisual3D.Content>
                <GeometryModel3D x:Name="Object3D">
                    <GeometryModel3D.Material>
                        <DiffuseMaterial Brush="Red"/>
                    </GeometryModel3D.Material>
                    <GeometryModel3D.Geometry>
                        <MeshGeometry3D
                            Positions="-0.25,0,1 -1,1,1 -1,-1,1 -0.25,-1,1 -0.25,0,1
                            -1,-1,1 0.25,0,1 1,-1,1 1,1,1 0.25,0,1 0.25,-1,1 1,-1,1
                            1,1,1 0,2,1 -1,1,1 -1,1,1 -0.25,0,1 0.25,0,1 1,1,1 1,1,-1
                            1,-1,-1 -1,-1,-1 -1,1,-1 1,1,-1 -1,1,-1 0,2,-1"
                            TriangleIndices="0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 15
                            17 18 19 20 21 19 21 22 23 24 25"/>
                    </GeometryModel3D.Geometry>
                </GeometryModel3D>
            </ModelVisual3D.Content>
        </ModelVisual3D>
    </Viewport3D.Children>
</Viewport3D>
```



Tensor product bezier surface

Movement directions: [ ] Direction X, [ ] Direction Y, [✓] Direction Z

Geometries to show: [✓] Control points, [✓] Control lines, [✓] Surface lines, [✓] Surface geometry, [✓] Smooth surface

Degree in direction U: 3

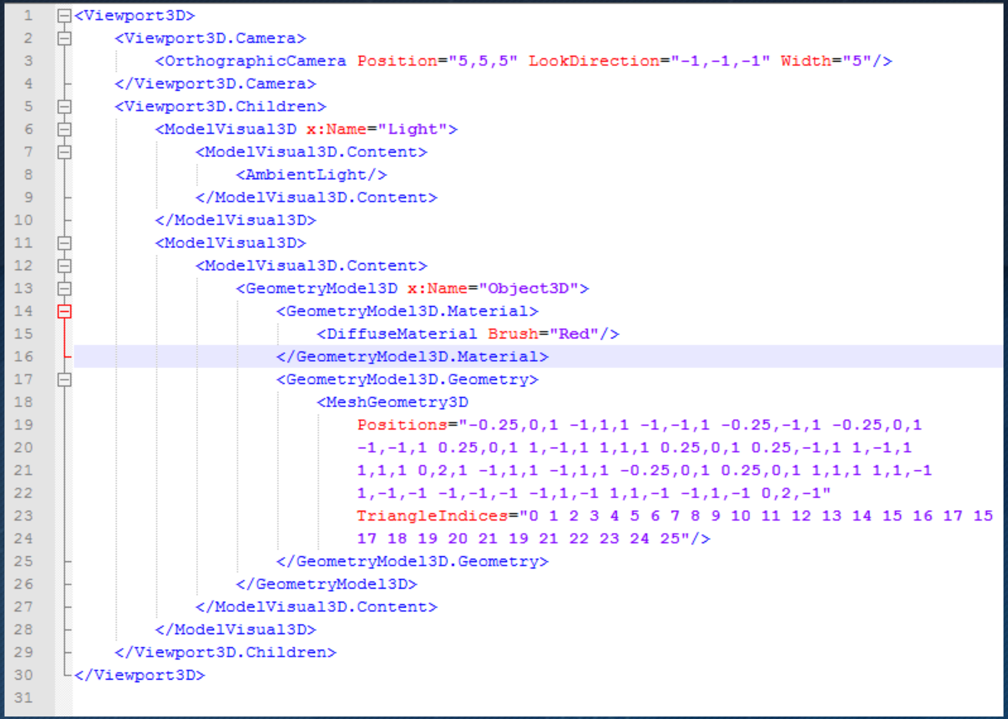
Devisions in direction U: 20

Degree in direction V: 3

Devisions in direction V: 20

- Built-in WPF functionality allowing to visualize 3D scene.
- Easy way to integrate 2D UIElements with 3D Viewport.
- 3D API follows familiar WPF patterns and conventions which makes it easy to use.
- 3D scene may be defined either with XAML or with procedural code.

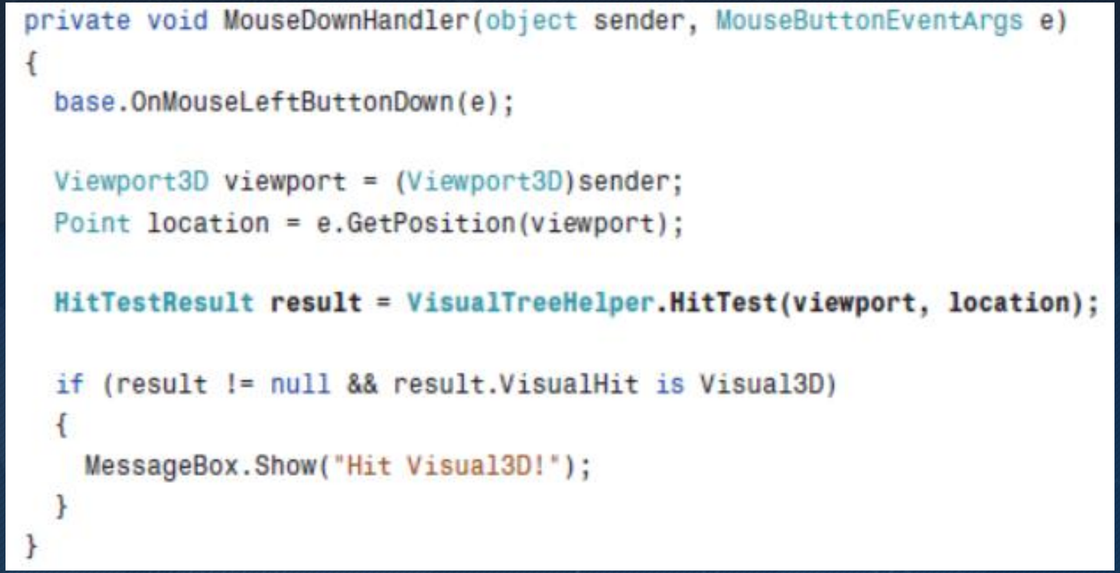
Scene graph hierarchy in WPF



```
<Viewport3D>
    <Viewport3D.Camera>
        <OrthographicCamera Position="5,5,5" LookDirection="-1,-1,-1" Width="5"/>
    </Viewport3D.Camera>
    <Viewport3D.Children>
        <ModelVisual3D x:Name="Light">
            <ModelVisual3D.Content>
                <AmbientLight/>
            </ModelVisual3D.Content>
        </ModelVisual3D>
        <ModelVisual3D>
            <ModelVisual3D.Content>
                <GeometryModel3D x:Name="Object3D">
                    <GeometryModel3D.Material>
                        <DiffuseMaterial Brush="Red"/>
                    </GeometryModel3D.Material>
                    <GeometryModel3D.Geometry>
                        <MeshGeometry3D
                            Positions="-0.25,0,1 -1,1,1 -1,-1,1 -0.25,-1,1 -0.25,0,1
                            -1,-1,1 0.25,0,1 1,-1,1 1,1,1 0.25,0,1 0.25,-1,1 1,-1,1
                            1,1,1 0,2,1 -1,1,1 -1,1,1 -0.25,0,1 0.25,0,1 1,1,1 1,1,-1
                            1,-1,-1 -1,-1,-1 -1,1,-1 1,1,-1 -1,1,-1 0,2,-1"
                            TriangleIndices="0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 15
                            17 18 19 20 21 19 21 22 23 24 25"/>
                    </GeometryModel3D.Geometry>
                </GeometryModel3D>
            </ModelVisual3D.Content>
        </ModelVisual3D>
    </Viewport3D.Children>
</Viewport3D>
```

- Viewport3D is the parent element
  - Viewport3D.Camera property
  - Viewport3D.Children property
    - Visual3D elements – define 3D objects with Model3D and position them with Transform3D
      - Model3D elements – define the 3D objects look with Geometry3D and Material
        - Geometry3D elements – define the geometry points, triangles, light normals and texture coordinates

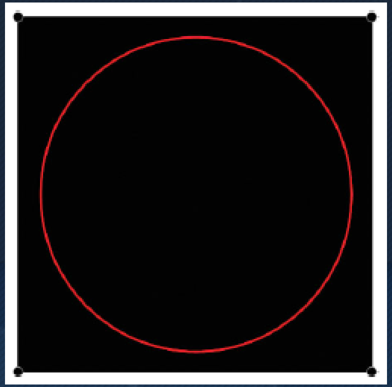
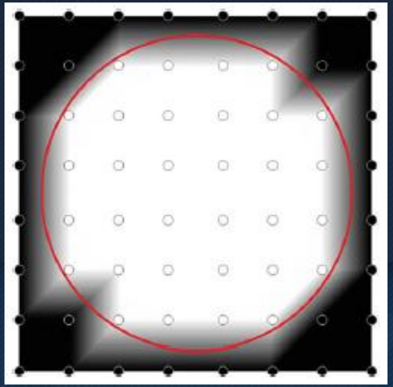
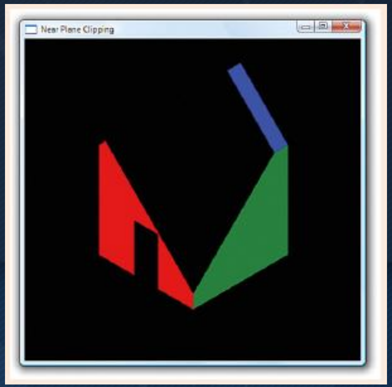
Interaction with 3D elements in WPF

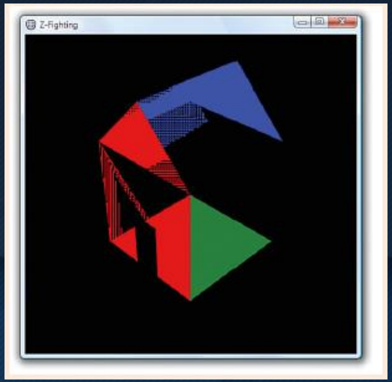


```
private void MouseDownHandler(object sender, MouseButtonEventArgs e)
{
  base.OnMouseLeftButtonDown(e);

  Viewport3D viewport = (Viewport3D)sender;
  Point location = e.GetPosition(viewport);

  HitTestResult result = VisualTreeHelper.HitTest(viewport, location);

  if (result != null && result.VisualHit is Visual3D)
  {
    MessageBox.Show("Hit Visual3D!");
  }
}
```

- Viewport3D
  - Attach to Mouse events of Viewport3D class.
  - Use HitTestResult to see if some Visual3D instance is interacted.
- UIElement3D – attach to mouse events directly to a single 3D object in the scene.
- Viewport2DVisual3D
  - Allows you to place 2D UIElement instances on the side of a 3D object.
  - Interact directly with the 2D UIElement instances using their mouse events

WPF 3D engine limitations

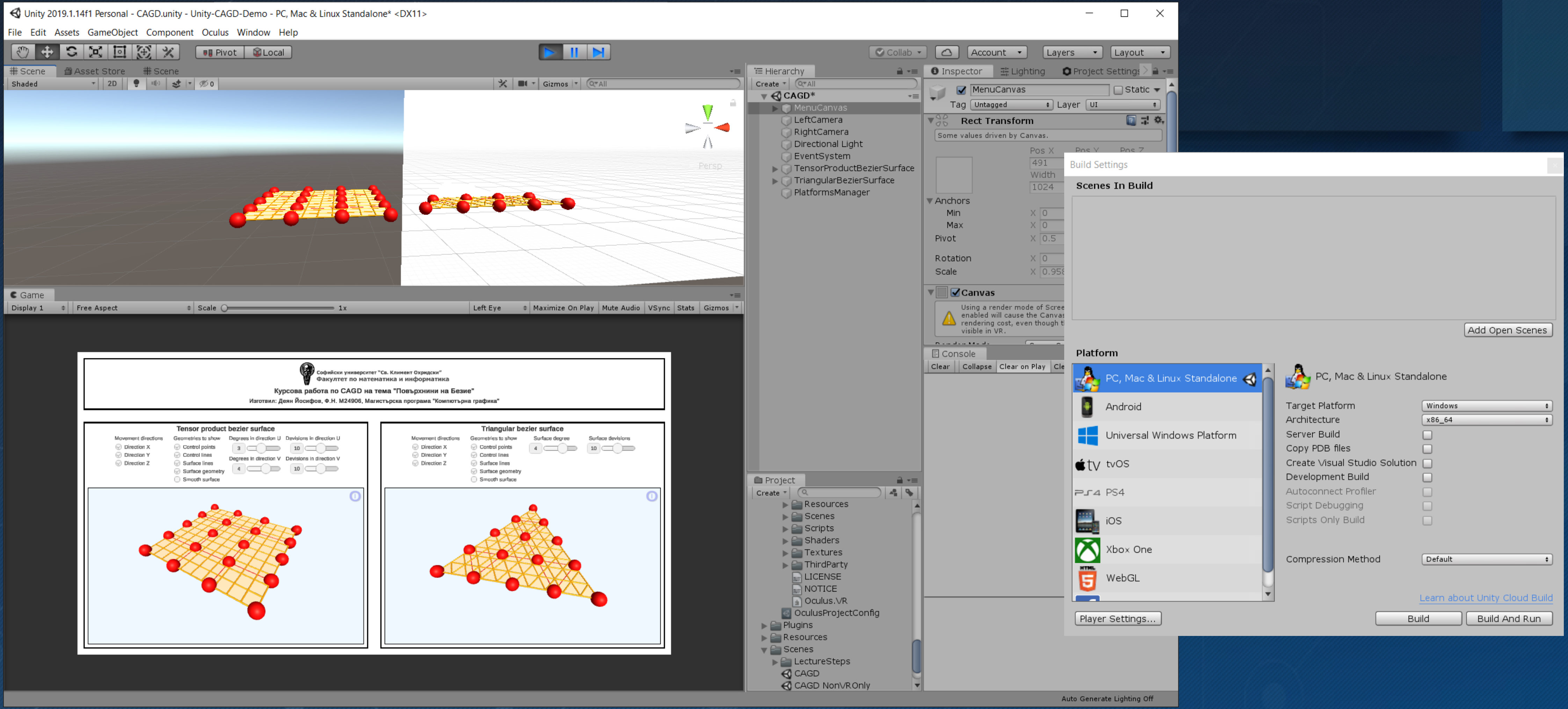


- SpotLight and PointLight
  - May not shade light some triangles.
  - May be workarounded in some scenarios by making more dense mesh.
- Transparency
  - The engine may not show some objects behind transparent triangles.
  - May be workarounded in some scenarios by reordering the children of the scene.
  - Translucent objects should be last scene's children to ensure that WPF engine will render them last.
- ProjectionCamera NearPlaneDistance and FarPlaneDistance properties may clip objects.
- Z-fighting
  - When surfaces are close to each other the camera cannot determine which to show.
  - May be workarounded by changing the FarPlaneDistance not to be Infinity.
  - Does not occur when surfaces coincide exactly – the second rendered will appear on top.

WPF 3D performance tips

- Viewport3D.IsHitTestVisible
  - Better be set to ‘false’ because otherwise every MouseMove makes hit tests with the whole scene.
  - Possible solution is to place Canvas over the Viewport3D and attach to Canvas Mouse events.
  - Still VisualTreeHelper.HitTest method may be used over Viewport3D when HitTestResult is needed.
- Freezable objects
  - Call Freeze method if they are not going to be changed.
  - Consider using this approach for Brush, Material and Geometry3D classes.
- Reuse whatever is repeating in the scene
  - Geometry3D (cylinders, cubes, spheres, ellipsoids, …)
  - Materials
  - GeometryModel3D
- Use only front materials on objects that are not intended to be seen from inside.
- Use simple materials when possible as material groups calculate the light on several passes.
- More tips may be found in this MSDN performance article.

Unity – designed for building cross platform 3D apps



Софийски университет "Св. Климент Охридски"
Факултет по математика и информатика

Курсова работа по CAGD на тема "Повърхнини на Безие"

Изготвил: Деян Йосифов, Ф.Н. М24906, Магистърска програма "Компютърна графика"

Build Settings

Scenes In Build

Add Open Scenes
Platform
PC, Mac & Linux Standalone
Android
Universal Windows Platform
tvOS
PS4
iOS
Xbox One
WebGL
PC, Mac & Linux Standalone

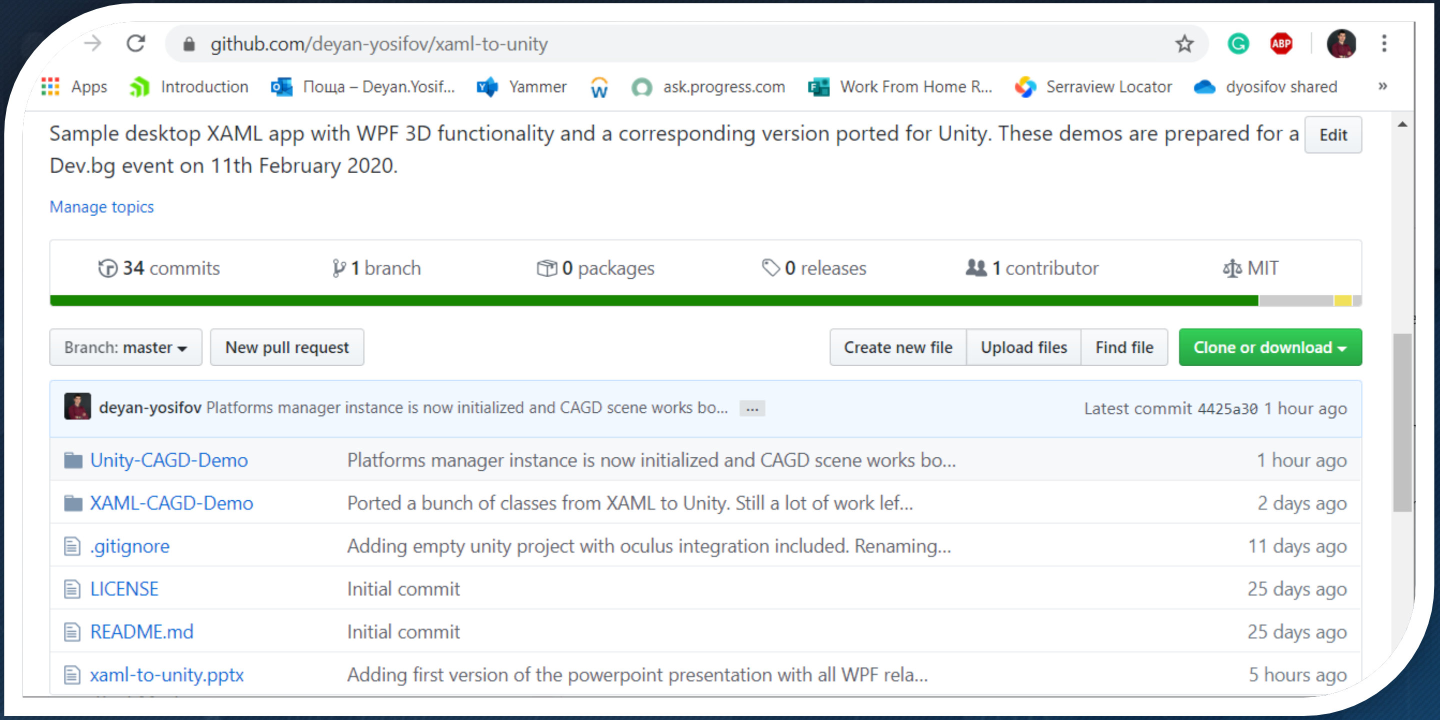
Target Platform	Windows
Architecture	x86_64
Server Build	☐
Copy PDB files	☐
Create Visual Studio Solution	☐
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐
Compression Method	Default

СЛЕДИ КАКВО ПРЕДСТОИ В ГРУПАТА НА DEV.BG/DOTNET

Unity vs WPF comparison

- Unity provides larger variety of options when it comes to defining 3D content – materials, shaders, camera postprocessing, etc.
- Unity uses composition pattern compared to the class inheritance which is typical for WPF.
- Unity provides flexible way to reuse and in the same time modify scene content through its prefabs and prefab variants functionalities.
- WPF is much more powerful when it comes to layouting 2D UI.
- Unity provides more optimized approach for handling and interacting with the 3D elements by separating the collision components from the mesh components.

Our mission today



github.com/deyan-yosifov/xaml-to-unity

Apps | Introduction | Поща – Deyan.Yosif... | Yammer | ask.progress.com | Work From Home R... | Serraview Locator | dyosifov shared

Sample desktop XAML app with WPF 3D functionality and a corresponding version ported for Unity. These demos are prepared for a Dev.bg event on 11th February 2020.

Manage topics

34 commits | 1 branch | 0 packages | 0 releases | 1 contributor | MIT

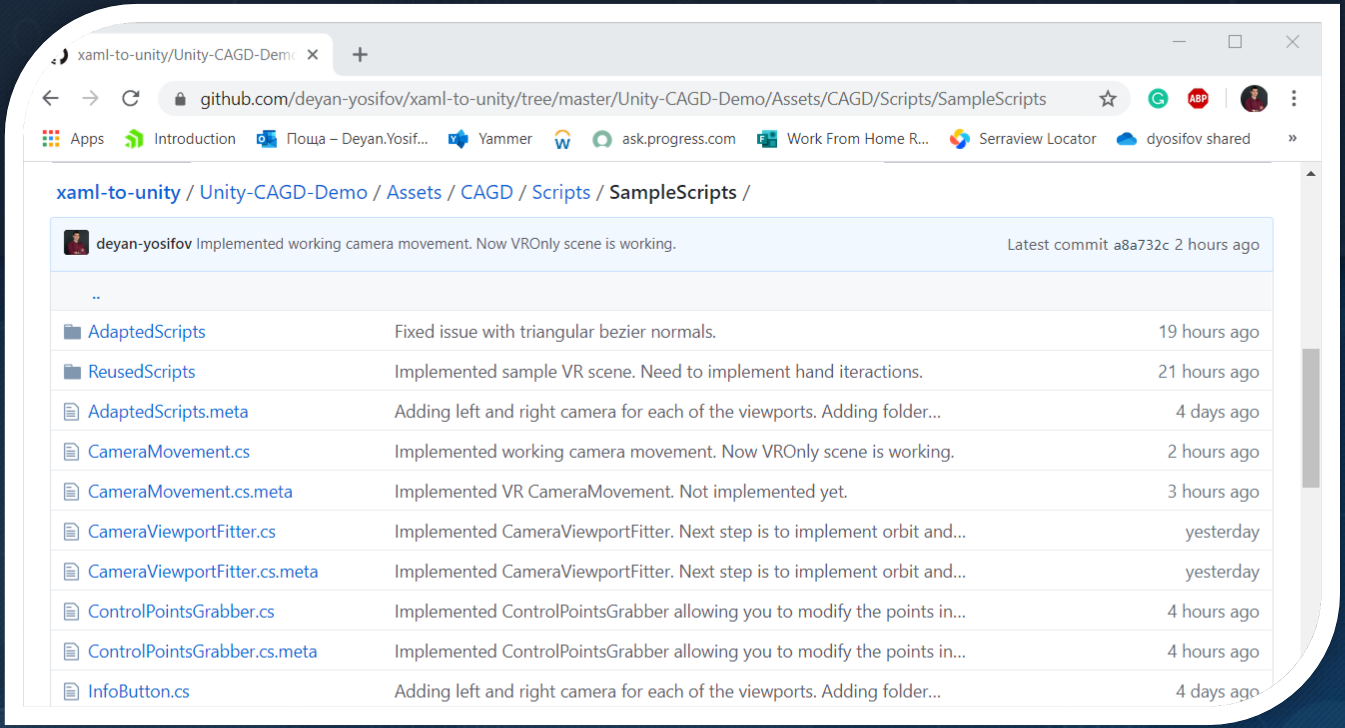
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

deyan-yosifov Platforms manager instance is now initialized and CAGD scene works bo... Latest commit 4425a30 1 hour ago

Unity-CAGD-Demo	Platforms manager instance is now initialized and CAGD scene works bo...	1 hour ago
XAML-CAGD-Demo	Ported a bunch of classes from XAML to Unity. Still a lot of work lef...	2 days ago
.gitignore	Adding empty unity project with oculus integration included. Renaming...	11 days ago
LICENSE	Initial commit	25 days ago
README.md	Initial commit	25 days ago
xaml-to-unity.pptx	Adding first version of the powerpoint presentation with all WPF rela...	5 hours ago

- Go to https://github.com/deyan-yosifov/xaml-to-unity
- See the WPF demo in XAML-CAGD-Demo folder.
- See how it can be converted to Unity in Unity-CAGD-Demo folder.
- Test our Unity demo for Web, Mobile, Desktop and VR.

Unity scripts demo hierarchy


xaml-to-unity/Unity-CAGD-Demo
github.com/deyan-yosifov/xaml-to-unity/tree/master/Unity-CAGD-Demo/Assets/CAGD/Scripts/SampleScripts
Apps
Introduction
Поща – Deyan.Yosif...
Yammer
ask.progress.com
Work From Home R...
Serraview Locator
dyosifov shared

xaml-to-unity / Unity-CAGD-Demo / Assets / CAGD / Scripts / SampleScripts /

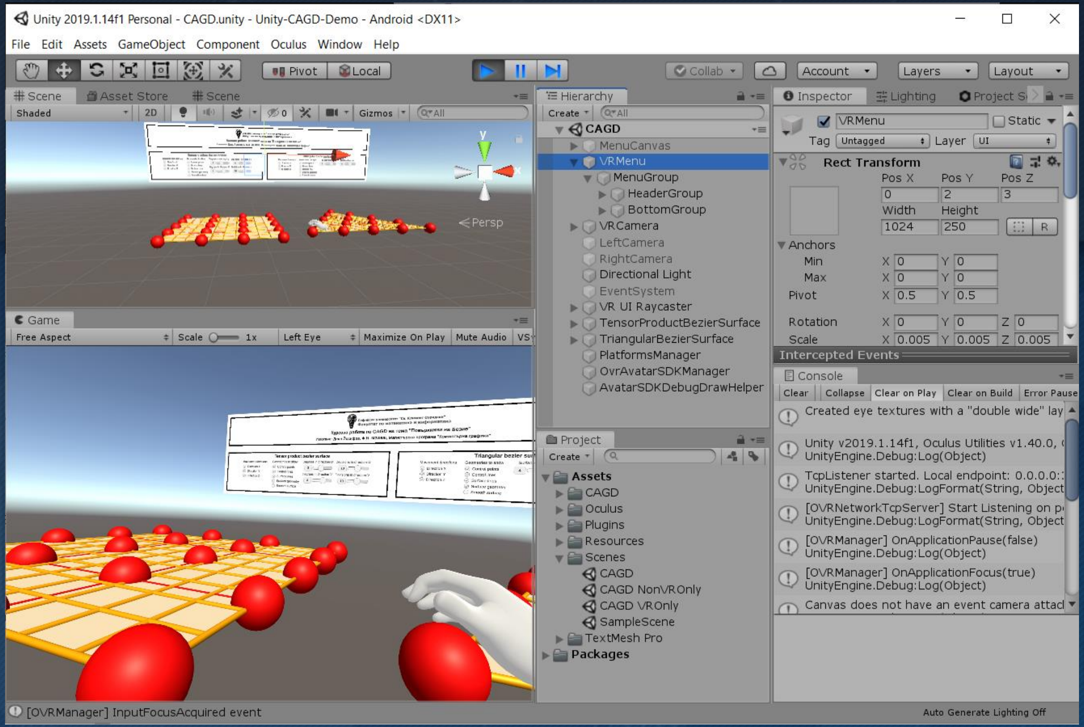
deyan-yosifov Implemented working camera movement. Now VROnly scene is working. Latest commit a8a732c 2 hours ago

..

AdaptedScripts	Fixed issue with triangular bezier normals.	19 hours ago
ReusedScripts	Implemented sample VR scene. Need to implement hand iteractions.	21 hours ago
AdaptedScripts.meta	Adding left and right camera for each of the viewports. Adding folder...	4 days ago
CameraMovement.cs	Implemented working camera movement. Now VROnly scene is working.	2 hours ago
CameraMovement.cs.meta	Implemented VR CameraMovement. Not implemented yet.	3 hours ago
CameraViewportFitter.cs	Implemented CameraViewportFitter. Next step is to implement orbit and...	yesterday
CameraViewportFitter.cs.meta	Implemented CameraViewportFitter. Next step is to implement orbit and...	yesterday
ControlPointsGrabber.cs	Implemented ControlPointsGrabber allowing you to modify the points in...	4 hours ago
ControlPointsGrabber.cs.meta	Implemented ControlPointsGrabber allowing you to modify the points in...	4 hours ago
InfoButton.cs	Adding left and right camera for each of the viewports. Adding folder...	4 days ago

- The Unity project contains CAGD scene which uses scripts from SampleScripts folder.
- ReusedScripts subfolder contains scripts which are practically reused from the XAML demo.
- AdaptedScripts subfolder has scripts which correspond to some XAML demo script but has more modification in order to adapt to Unity technology requirements.
- The rest scripts are Unity specific.

Demo time



VRMenu Static
Tag Untagged Layer UI

	Pos X	Pos Y	Pos Z
	0	2	3
	Width	Height	
	1024	250	
Anchors			
Min	X 0	Y 0	
Max	X 0	Y 0	
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 0.005	Y 0.005	Z 0.005

```
Created eye textures with a "double wide" lay
Unity v2019.1.14f1, Oculus Utilities v1.40.0, (
UnityEngine.Debug:Log(Object)
TcpListener started. Local endpoint: 0.0.0.0::
UnityEngine.Debug:LogFormat(String, Object
[OVRNetworkTcpServer] Start Listening on p
UnityEngine.Debug:LogFormat(String, Object
[OVRManager] OnApplicationPause(false)
UnityEngine.Debug:Log(Object)
[OVRManager] OnApplicationFocus(true)
UnityEngine.Debug:Log(Object)
Canvas does not have an event camera attacl
```

```
[OVRManager] InputFocusAcquired event
```

- Let’s build our demo Unity scene so that it works for Web, Mobile, Desktop and VR.
- A web version of the app is already uploaded and can be seen on: http://deyan-yosifov.com/devbg/CAGD/

Thank you!

Контакти:

LinkedIn: https://www.linkedin.com/in/deyan-yosifov/
Github: https://github.com/deyan-yosifov
Twitter: https://twitter.com/deyan_yosifov

Следващо събитие:

Bootstrapping microservices in seconds