Manually separate trunk during vegetation optimization

2022-04-04

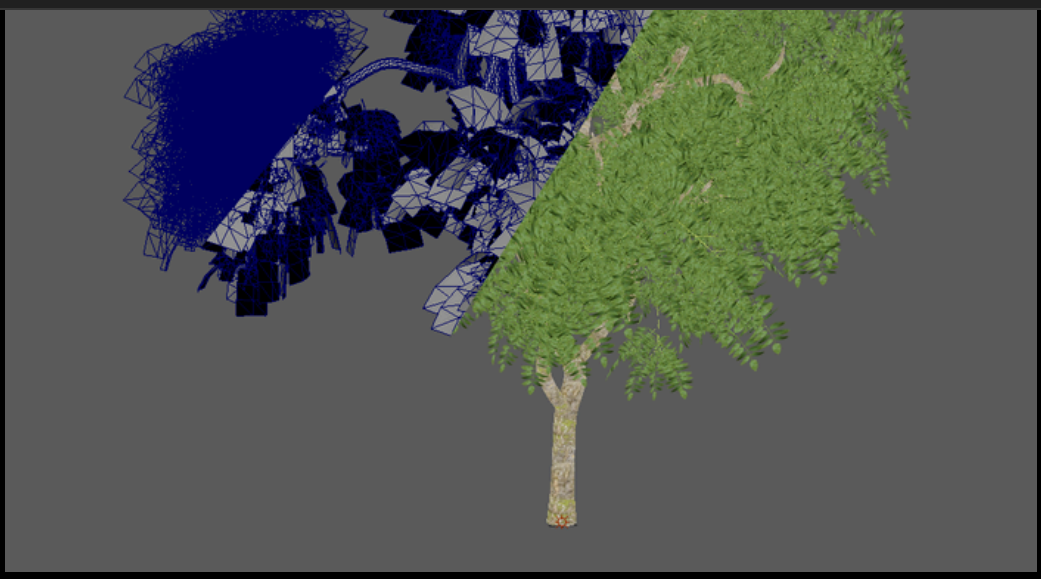
Material Casting | Billboard cloud | Color caster | Python | Selection Set

Written by Jesper Tingvall, Product Expert, Simplygon

Disclaimer: *The code in this post is written using version 9.2.4200.0 of Simplygon and Maya 2022. If you encounter this post at a later stage, some of the API calls might have changed. However, the core concepts should still remain valid.*

Introduction

Simplygon has a pipeline specific tailored for vegetation optimization; Billboard Cloud for Vegetation. One of its features is an automatic trunk detector which can separate out the trunk from all leaves and run it in a separate reduction processor. However for this blog post we are going to do that manually to increase our control over the processing.



Prerequisites

This example will use the Simplygon integration in Maya, but the same concepts can be applied to all other integrations of the Simplygon API.

Problem to solve

We want to optimize a vegetation asset using the billboard cloud for vegetation pipeline. However we want to have more control over what is counted as trunk and what is counted as leaves. One reason for this could be that it is hard to find seperation values that works for all of our assets.

Solution

The solution is to manually split the asset into two parts; leaves and trunk. We can then process those parts seperate using the pipelines we want.

Export selection

First we are going to export selection from Maya into a temporary file.

```
    scene = sg.CreateScene()
    scene.LoadFromFile(tmp_file)
    return scene
```

Split tree

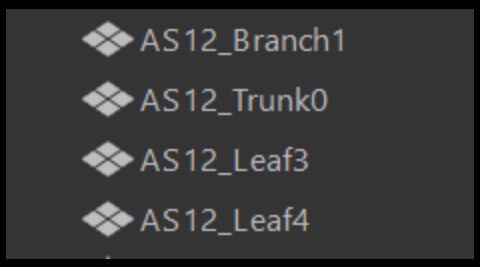
The asset will be split into two different different Selection Sets; one for the trunk and one for leaves and tiny branches.

```
# Split scene into 2 selection sets; leaves and trunk
leaf_set = sg.CreateSelectionSet()
trunk_set = sg.CreateSelectionSet()
leaf_set_id = scene.GetSelectionSetTable().AddSelectionSet(leaf_
trunk_set_id = scene.GetSelectionSetTable().AddSelectionSet(trur
split_tree(scene, scene.GetRootNode(), leaf_set, trunk_set)
```

We will split the scene recursively into the two sets depending on if the node is considered to be part of the trunk or not.

```
def split_tree(scene, scene_node, leaf_set, trunk_set):
    """Splits scene_node recursively into two sets; leaf_set and
    for i in range(0, scene_node.GetChildCount()):
        child = scene_node.GetChild(i)
        if child.IsA("ISceneMesh"):
            scene_mesh = Simplygon.spSceneMesh.SafeCast(child)
            if is_mesh_trunk(scene, scene_mesh):
                trunk_set.AddItem(child.GetNodeGUID())
            else:
                leaf_set.AddItem(child.GetNodeGUID())

        split_tree(scene, child,  leaf_set, trunk_set)
```

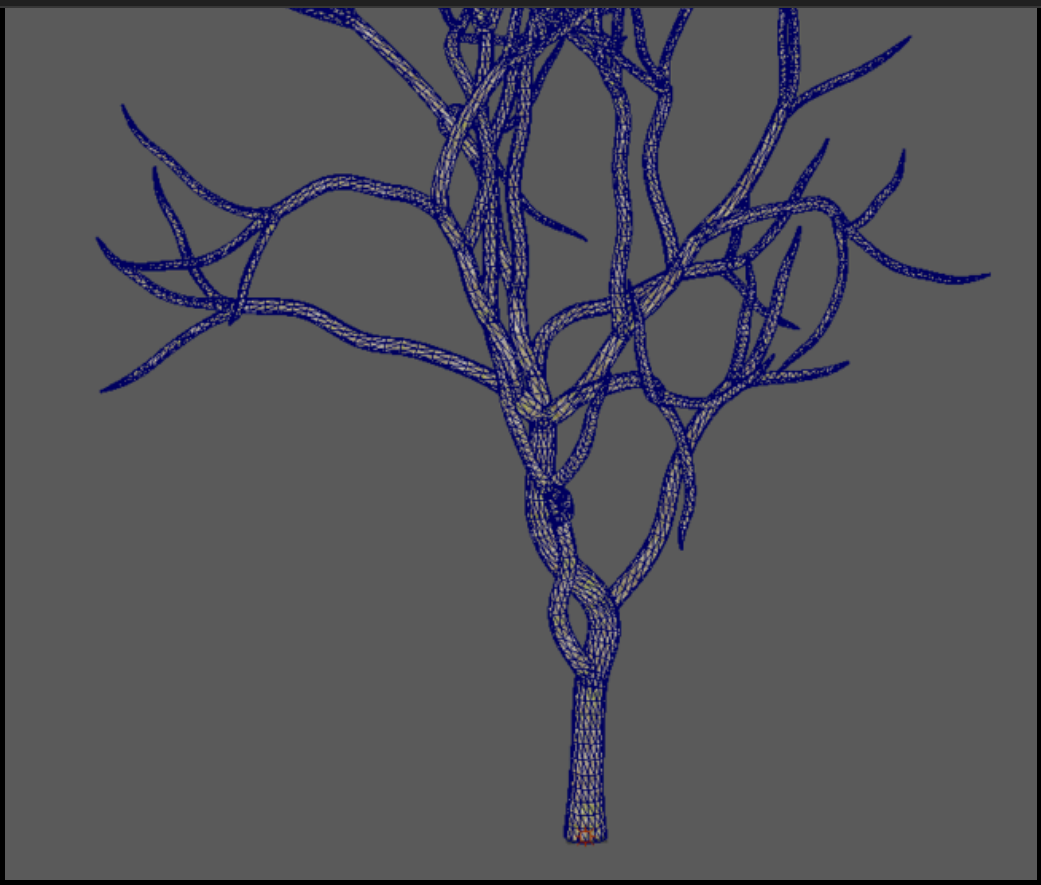
material with a specific name. In this assets case however we are going to use the most basic case; the name of the mesh.



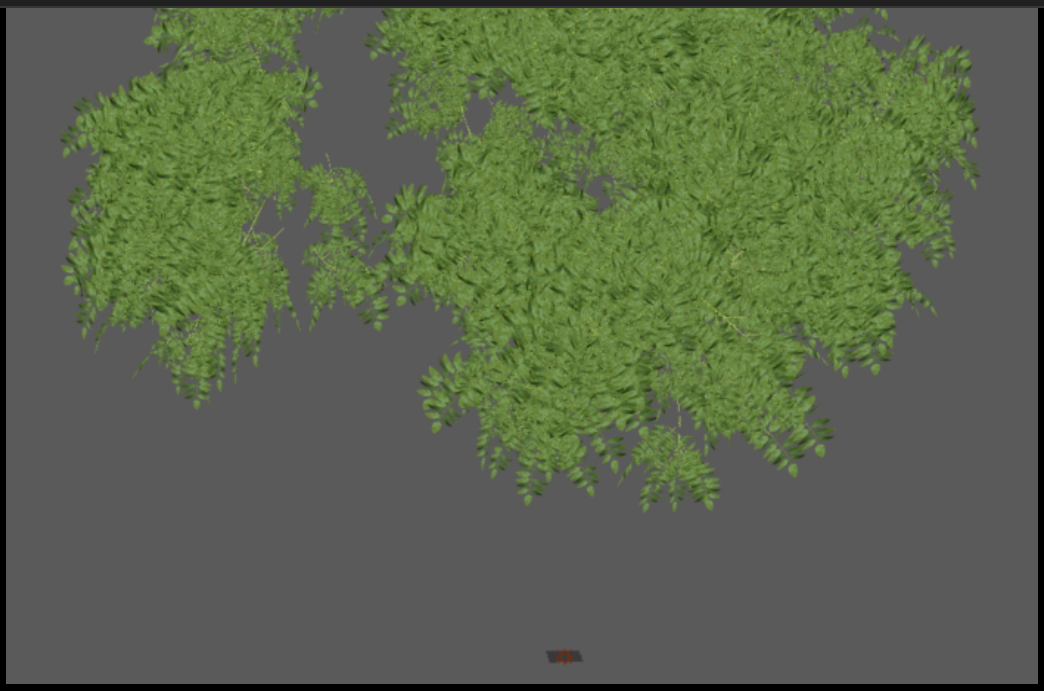
An inspection of the asset reveals that the part we want to process with the reducer contains Trunk. Branches and leaves should be processed with the billboard cloud processor.

```
def is_mesh_trunk(scene, scene_mesh):
    """Returns True if scene_mesh is trunk or False if it is lea
    return "trunk" in scene_mesh.GetName().lower()
```

After splitting the scene here is the trunk.



And here are the leaves including small branches.



Reduce trunk pipeline

The trunk will be reduced using a reduction pipeline. We are going to use triangle ratio as target.

```
def create_reduction_pipeline(sg, set_id):
    """Returns a reduction pipeline for trunk"""

    pipeline = sg.CreateReductionPipeline()
    pipeline_settings = pipeline.GetReductionSettings()

    pipeline_settings.SetReductionTargetTriangleRatio(0.10)
    pipeline_settings.SetReductionTargetTriangleRatioEnabled(Tru
    return pipeline
```

After processing this is the resulting trunk.

Billboard cloud for leaves pipeline

For leaves and tiny branches we are going to use Billboard Cloud for vegetation pipeline. We set `SetSeparateTrunkAndFoilage` to `False` since we are separating it manually.

```
def create_billboard_cloud_pipeline(sg, set_id):
    """Returns a billboard cloud for vegetation pipeline for lea

    # Create the Impostor processor.
    sgBillboardCloudVegetationPipeline = sg.CreateBillboardCloud
    sgBillboardCloudSettings = sgBillboardCloudVegetationPipelin
    sgMappingImageSettings = sgBillboardCloudVegetationPipeline.
    sgMappingImageSettings.SetTexCoordName("MaterialLOD") # Need

    # Set billboard cloud mode to Foliage and settings.
    sgBillboardCloudSettings.SetBillboardMode( Simplygon.EBillbo
```

```
# Do not seperate Trunk and Foilage.
sgFoliageSettings.SetSeparateTrunkAndFoliage( False )

# Setting the size of the output material for the mapping ir
sgMappingImageSettings.SetMaximumLayers( 3 )
sgOutputMaterialSettings = sgMappingImageSettings.GetOutputM
sgOutputMaterialSettings.SetTextureWidth( 1024 )
sgOutputMaterialSettings.SetTextureHeight( 1024 )
sgOutputMaterialSettings.SetMultisamplingLevel( 2 )
```

Firstly we are going to add a Color Caster for the diffuse channel and set Maya specific settings. One thing to look out for in particular is `SetOutputSRGB`. If your casted texture differs a little bit in color from the original it is quite likely it is incorrect.

```
# Add diffuse material caster to pipeline and set up with Ma
print("Add diffuse material caster to pipeline.")
sgDiffuseCaster = sg.CreateColorCaster()
sgDiffuseCasterSettings = sgDiffuseCaster.GetColorCasterSett
sgDiffuseCasterSettings.SetMaterialChannel( "color" )
sgDiffuseCasterSettings.SetOutputImageFileFormat( Simplygon.
sgDiffuseCasterSettings.SetBakeOpacityInAlpha( False )
sgDiffuseCasterSettings.SetOutputPixelFormat( Simplygon.EPix
sgDiffuseCasterSettings.SetDilation( 10 )
sgDiffuseCasterSettings.SetFillMode( Simplygon.EAtlasFillMod
sgDiffuseCasterSettings.SetUseMultisampling(True)
sgDiffuseCasterSettings.SetOutputSRGB(True)
sgDiffuseCasterSettings.SetOpacityChannel("transparency")
sgBillboardCloudVegetationPipeline.AddMaterialCaster( sgDiff
```

We add a Color Caster for the specular channel and add Maya specific settings.

```
# Add specular material caster to pipeline and set up with M
print("Add specular material caster to pipeline.")
sgSpecularCaster = sg.CreateColorCaster()
```

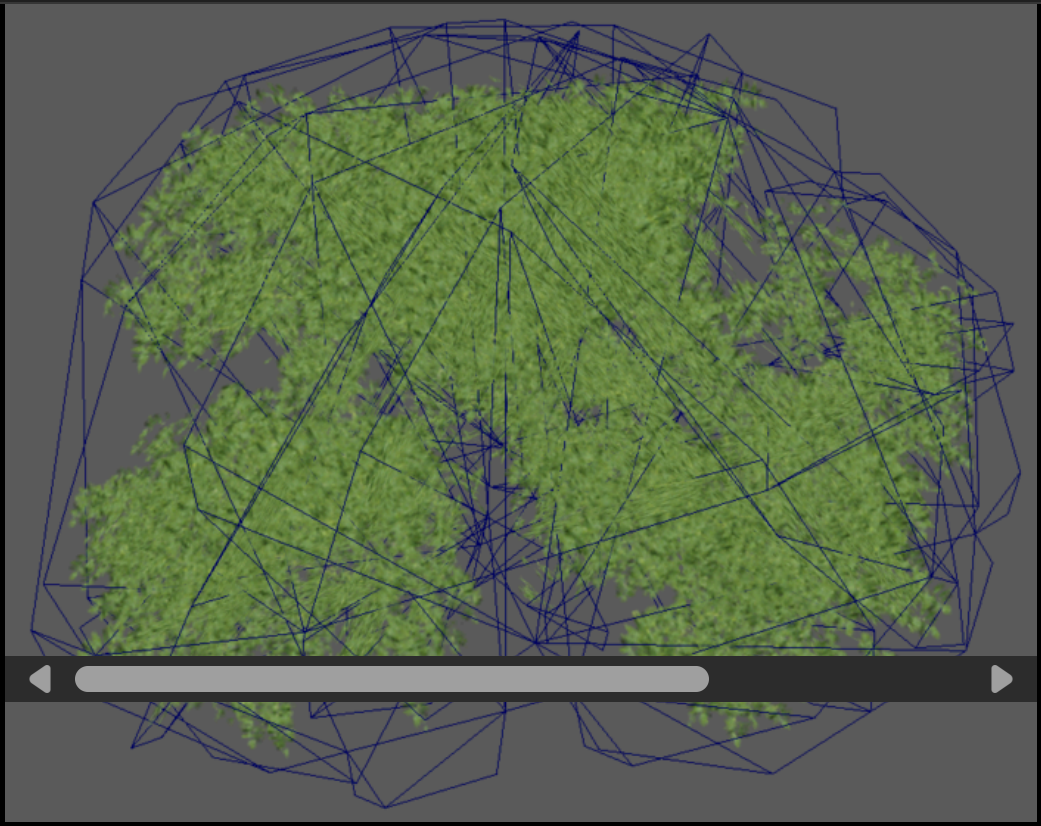
```
sgBillboardCloudVegetationPipeline.AddMaterialCaster( sgSpec
```

We add a Normal Caster so we can bake a normal map for the leaves and set corresponding Maya settings.

```
# Add normals material caster to pipeline and set up with Ma
print("Add normals material caster to pipeline.")
sgNormalsCaster = sg.CreateNormalCaster()
sgNormalsCasterSettings = sgNormalsCaster.GetNormalCasterSet
sgNormalsCasterSettings.SetMaterialChannel( "normalCamera" )
sgNormalsCasterSettings.SetGenerateTangentSpaceNormals( True
sgNormalsCasterSettings.SetOutputImageFileFormat( Simplygon.
sgNormalsCasterSettings.SetDilation( 10 )
sgNormalsCasterSettings.SetFillMode( Simplygon.EAtlasFillMod
sgBillboardCloudVegetationPipeline.AddMaterialCaster( sgNorm
```

Lastly we are going to add an Opacity Caster for casting the transparency and set it up for Maya.

```
# Add opacity material casting to pipelineand set up with Ma
print("Add opacity material casting to pipeline.")
sgOpacityCaster = sg.CreateOpacityCaster()
sgOpacityCasterSettings = sgOpacityCaster.GetOpacityCasterSe
sgOpacityCasterSettings.SetMaterialChannel( "transparency" )
sgOpacityCasterSettings.SetOpacityChannel("transparency")
sgOpacityCasterSettings.SetOutputImageFileFormat( Simplygon.
sgOpacityCasterSettings.SetFillMode( Simplygon.EAtlasFillMod
sgOpacityCasterSettings.SetDilation( 0 )
sgOpacityCasterSettings.SetUseMultisampling(True)
sgOpacityCasterSettings.SetOutputPixelFormat( Simplygon.EPix
sgOpacityCasterSettings.SetOpacityChannelComponent(Simplygon
sgOpacityCasterSettings.SetOutputOpacityType(Simplygon.EOpac
sgOpacityCasterSettings.SetOutputSRGB(False)
sgBillboardCloudVegetationPipeline.AddMaterialCaster( sgOpac
```



Putting it all together

We start by exporting the selected asset from Maya and splitting it up into our two selection sets; `leaf_set` and `trunk_set`.

```
def process_selection(sg):
    """Optimize Maya selected vegetation asset."""

    # Set tangent space for Maya
    sg.SetGlobalDefaultTangentCalculatorTypeSetting(Simplygon.ET

    scene = export_selection(sg)

    # Split scene into 2 selection sets; leaves and trunk
    leaf_set = sg.CreateSelectionSet()
    trunk_set = sg.CreateSelectionSet()
    leaf_set_id = scene.GetSelectionSetTable().AddSelectionSet(1
```

scene. Then we are going to remove the trunk and leaves from each scene via `RemoveSceneNodesInSelectionSet`. The result is two scenes; one with trunk and one with leaves and tiny branches.

```
# Create a scene containing only the leaves
leaf_scene = scene.NewCopy()
leaf_scene.RemoveSceneNodesInSelectionSet(trunk_set_id)

# Remove all leaves from original scene so we are left with
scene.RemoveSceneNodesInSelectionSet(leaf_set_id)
```

We process our trunk scene using our reduction pipeline and our leaf scene using out billboard cloud for vegetation pipeline.

```
# Process trunk
reduction_pipeline = create_reduction_pipeline(sg, 0)
reduction_pipeline.RunScene(scene, Simplygon.EPipelineRunMo

# Process leaves
leaf_pipeline = create_billboard_cloud_pipeline(sg, 0)
leaf_pipeline.RunScene(leaf_scene, Simplygon.EPipelineRunMo
```

Lastly we are going to merge the two processed scenes together using `Append` which copies over everything from `leaf_scene` into `scene`.

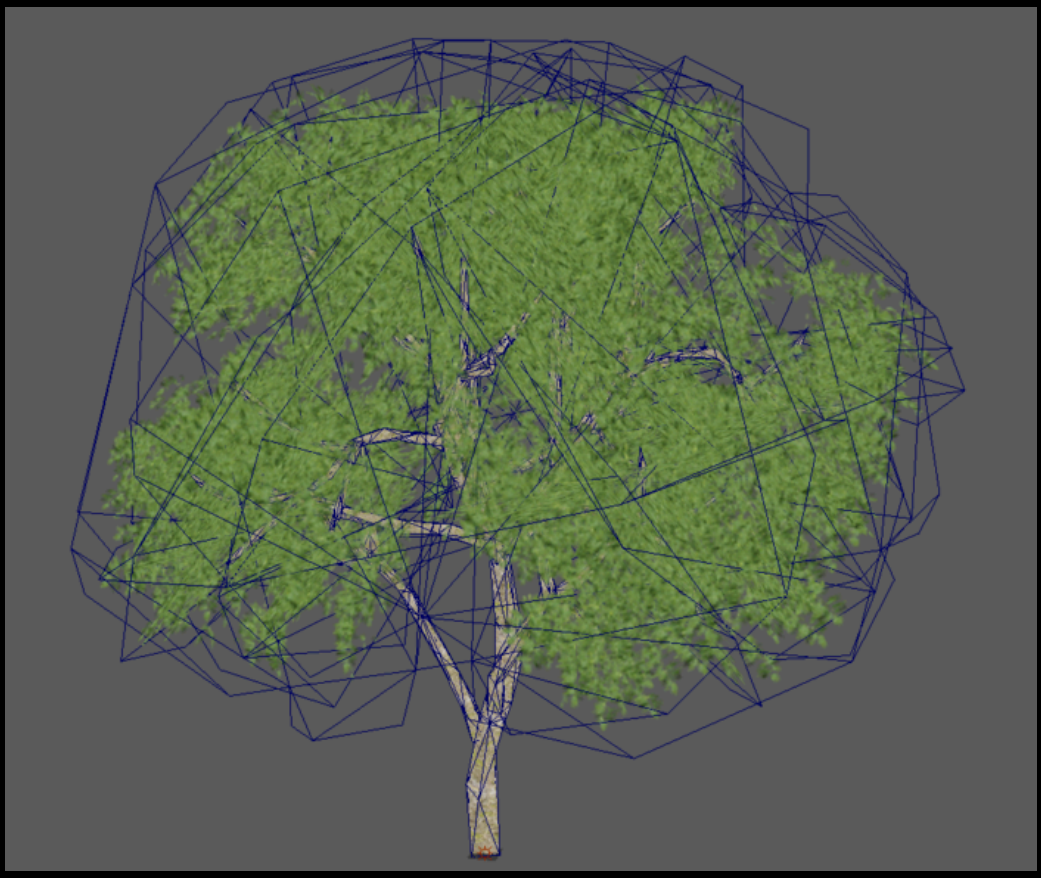
```
# Add processed leaves scene to processed trunk scene
scene.Append(leaf_scene)

# Export to Maya
import_results(scene)
```

Import selection

```python
def import_results(scene):
    """Import the Simplygon scene into Maya."""
    scene.SaveToFile(tmp_file)
    cmds.Simplygon(imp=tmp_file, lma=True)
```

Result

The result is a highly optimized tree and we have total control over which parts that are reduced or turned into a billboard.



Asset	Verts	Materials
Original asset	197 872	4
Optimized asset	1 144	2

One benefit of this approach is that we can process the trunk with another pipeline if we want to. We also get access to more options in the

```
# Copyright (c) Microsoft Corporation.
# Licensed under the MIT license.

from simplygon import simplygon_loader
from simplygon import Simplygon
import maya.cmds as cmds
import os

tmp_file = "c:/Temp/export.sb"

def export_selection(sg):
    """Export the current selected objects into a Simplygon scer
    cmds.Simplygon(exp = tmp_file)
    scene = sg.CreateScene()
    scene.LoadFromFile(tmp_file)
    return scene


def import_results(scene):
    """Import the Simplygon scene into Maya."""
    scene.SaveToFile(tmp_file)
    cmds.Simplygon(imp=tmp_file, lma=True)

def split_tree(scene, scene_node, leaf_set, trunk_set):
    """Splits scene_node recursively into two sets; leaf_set and
    for i in range(0, scene_node.GetChildCount()):
        child = scene_node.GetChild(i)
        if child.IsA("ISceneMesh"):
            scene_mesh = Simplygon.spSceneMesh.SafeCast(child)
            if is_mesh_trunk(scene, scene_mesh):
                trunk_set.AddItem(child.GetNodeGUID())
            else:
                leaf_set.AddItem(child.GetNodeGUID())

        split_tree(scene, child,  leaf_set, trunk_set)


def is_mesh_trunk(scene, scene_mesh):
    """Returns True if scene_mesh is trunk or False if it is lea
```

```
    pipeline_settings = pipeline.GetReductionSettings()

    pipeline_settings.SetReductionTargetTriangleRatio(0.10)
    pipeline_settings.SetReductionTargetTriangleRatioEnabled(Tru
    return pipeline


def create_billboard_cloud_pipeline(sg, set_id):
    """Returns a billboard cloud for vegetation pipeline for lea

    # Create the Impostor processor.
    sgBillboardCloudVegetationPipeline = sg.CreateBillboardCloud
    sgBillboardCloudSettings = sgBillboardCloudVegetationPipelin
    sgMappingImageSettings = sgBillboardCloudVegetationPipeline.
    sgMappingImageSettings.SetTexCoordName("MaterialLOD") # Need

    # Set billboard cloud mode to Foliage and settings.
    sgBillboardCloudSettings.SetBillboardMode( Simplygon.EBillbo
    sgBillboardCloudSettings.SetBillboardDensity( 0.4 )
    sgBillboardCloudSettings.SetGeometricComplexity( 0.5 )
    sgBillboardCloudSettings.SetMaxPlaneCount( 10 )
    sgBillboardCloudSettings.SetTwoSided( True )
    sgFoliageSettings = sgBillboardCloudSettings.GetFoliageSetti

    # Do not seperate Trunk and Foilage.
    sgFoliageSettings.SetSeparateTrunkAndFoliage( False )

    # Setting the size of the output material for the mapping im
    sgMappingImageSettings.SetMaximumLayers( 3 )
    sgOutputMaterialSettings = sgMappingImageSettings.GetOutputM
    sgOutputMaterialSettings.SetTextureWidth( 1024 )
    sgOutputMaterialSettings.SetTextureHeight( 1024 )
    sgOutputMaterialSettings.SetMultisamplingLevel( 2 )

    # Add diffuse material caster to pipeline and set up with Ma
    print("Add diffuse material caster to pipeline.")
    sgDiffuseCaster = sg.CreateColorCaster()
    sgDiffuseCasterSettings = sgDiffuseCaster.GetColorCasterSett
    sgDiffuseCasterSettings.SetMaterialChannel( "color" )
    sgDiffuseCasterSettings.SetOutputImageFileFormat( Simplygon.
```

```
sgDiffuseCasterSettings.SetOpacityChannel("transparency")
sgBillboardCloudVegetationPipeline.AddMaterialCaster( sgDif

# Add specular material caster to pipeline and set up with M
print("Add specular material caster to pipeline.")
sgSpecularCaster = sg.CreateColorCaster()
sgSpecularCasterSettings = sgSpecularCaster.GetColorCasterSe
sgSpecularCasterSettings.SetMaterialChannel( "specularColor"
sgSpecularCasterSettings.SetOutputImageFileFormat( Simplygon
sgSpecularCasterSettings.SetDilation( 0 )
sgSpecularCasterSettings.SetFillMode( Simplygon.EAtlasFillMo
sgSpecularCasterSettings.SetOpacityChannel("transparency")
sgBillboardCloudVegetationPipeline.AddMaterialCaster( sgSpec

# Add normals material caster to pipeline and set up with Ma
print("Add normals material caster to pipeline.")
sgNormalsCaster = sg.CreateNormalCaster()
sgNormalsCasterSettings = sgNormalsCaster.GetNormalCasterSet
sgNormalsCasterSettings.SetMaterialChannel( "normalCamera" )
sgNormalsCasterSettings.SetGenerateTangentSpaceNormals( True
sgNormalsCasterSettings.SetOutputImageFileFormat( Simplygon.
sgNormalsCasterSettings.SetDilation( 10 )
sgNormalsCasterSettings.SetFillMode( Simplygon.EAtlasFillMod
sgBillboardCloudVegetationPipeline.AddMaterialCaster( sgNorm

# Add opacity material casting to pipelineand set up with Ma
print("Add opacity material casting to pipeline.")
sgOpacityCaster = sg.CreateOpacityCaster()
sgOpacityCasterSettings = sgOpacityCaster.GetOpacityCasterSe
sgOpacityCasterSettings.SetMaterialChannel( "transparency" )
sgOpacityCasterSettings.SetOpacityChannel("transparency")
sgOpacityCasterSettings.SetOutputImageFileFormat( Simplygon.
sgOpacityCasterSettings.SetFillMode( Simplygon.EAtlasFillMod
sgOpacityCasterSettings.SetDilation( 0 )
sgOpacityCasterSettings.SetUseMultisampling(True)
sgOpacityCasterSettings.SetOutputPixelFormat( Simplygon.EPix
sgOpacityCasterSettings.SetOpacityChannelComponent(Simplygon
sgOpacityCasterSettings.SetOutputOpacityType(Simplygon.EOpac
sgOpacityCasterSettings.SetOutputSRGB(False)
```

```
def process_selection(sg):
    """Optimize Maya selected vegetation asset."""

    # Set tangent space for Maya
    sg.SetGlobalDefaultTangentCalculatorTypeSetting(Simplygon.ET

    scene = export_selection(sg)

    # Split scene into 2 selection sets; leaves and trunk
    leaf_set = sg.CreateSelectionSet()
    trunk_set = sg.CreateSelectionSet()
    leaf_set_id = scene.GetSelectionSetTable().AddSelectionSet(l
    trunk_set_id = scene.GetSelectionSetTable().AddSelectionSet(
    split_tree(scene, scene.GetRootNode(), leaf_set, trunk_set)

    # Create a scene containing only the leaves
    leaf_scene = scene.NewCopy()
    leaf_scene.RemoveSceneNodesInSelectionSet(trunk_set_id)

    # Remove all leaves from original scene so we are left with
    scene.RemoveSceneNodesInSelectionSet(leaf_set_id)

    # Process trunk
    reduction_pipeline = create_reduction_pipeline(sg, 0)
    reduction_pipeline.RunScene(scene, Simplygon.EPipelineRunMod

    # Process leaves
    leaf_pipeline = create_billboard_cloud_pipeline(sg, 0)
    leaf_pipeline.RunScene(leaf_scene, Simplygon.EPipelineRunMod

    # Add processed leaves scene to processed trunk scene
    scene.Append(leaf_scene)

    # Export to Maya
    import_results(scene)

def main():
    sg = simplygon_loader.init_simplygon()
    process_selection(sg)
    del sg
```

Blog

Search in all posts Search

Categories

3D Scans | 3ds Max | Adobe Substance 3D | Aggregation | Announcements | Batching | Best practices | Blender | Bone reduction | Characters | Customer story | Events | Foliage | Houdini | Impostor | Interior Design | Kitbashing | LOD0 Optimization | Maya | Open world | Physics meshes | Porting | Reduction | Remeshing | Scripting | Troubleshooting | Unity | Unreal Engine

Tags

Automation | bary | Billboard cloud | C# | C++ | Cascaded pipelines

Material Casting | MaterialProxyReplace | Micro-Mesh | Modular seams | Multisampling | Normal Map | Observer | Outer shell | Pipelines | Processor | Python | Quad reduction | Scene descriptor | Scene Validator | Seasonal | Selection Set | Shaders | Shading Network | Simplygon 10.0 | Simplygon 10.1 | Simplygon 10.2 | Simplygon 9.0 | Simplygon 9.1 | Simplygon 9.2 | SkinnedMeshRenderer | sRGB | Standin | Surface mapper | Tessellated attributes | Transparency | Unity HDRP | Unity URP | Unreal Engine 4.25 | Unreal Engine 5 | Unreal Engine 5.1 | Unreal Engine 5.2 | User fields | Vertex color caster | Vertex colors | Vertex lock | Vertex weight | Visibility Culling | WebGL