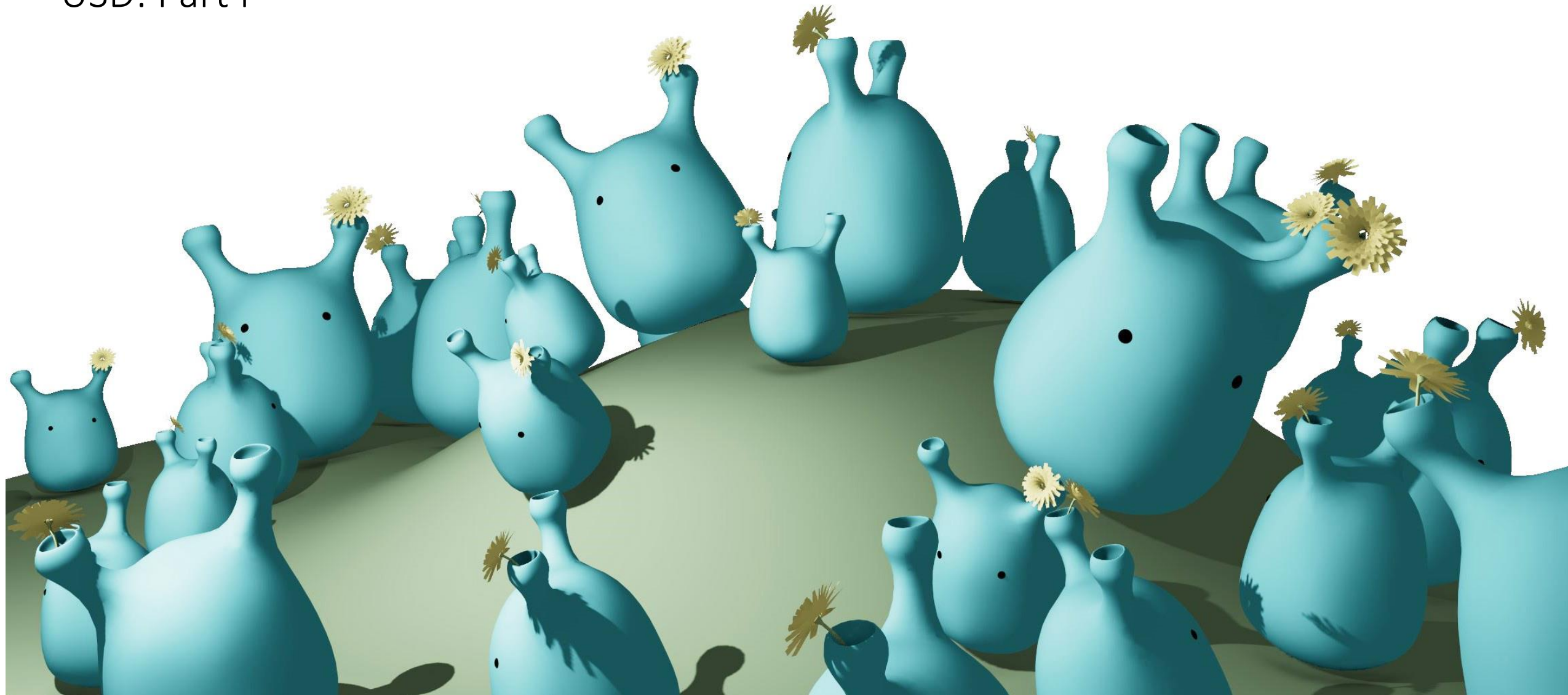


Bifrost Workshop

Week8

USD: Part I



USD in Bifrost Series:

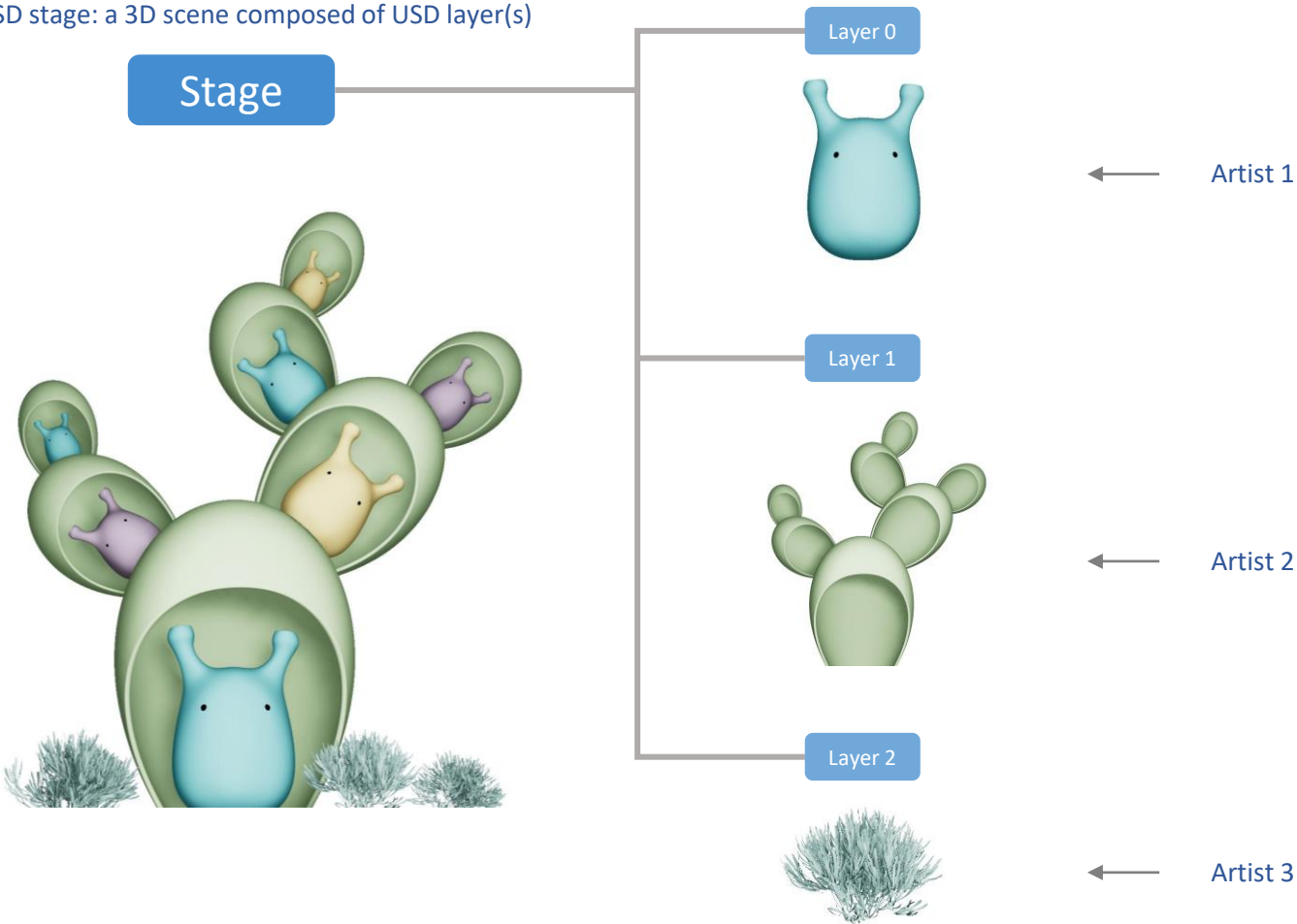
<https://youtube.com/playlist?list=PL8hZ6hQCGHMOVPTLY8J-yeBE3TD0dEd8Z4>

What is USD?

USD: Universal Scene Description

USD is a framework developed by Pixar Animation Studios to increase pipeline efficiency and facilitate collaborations. It provides a common language for defining, assembling, and editing data. It allows artists to simultaneously work on the same scene, make changes as they need and try out different compositions without changing other artists' work.

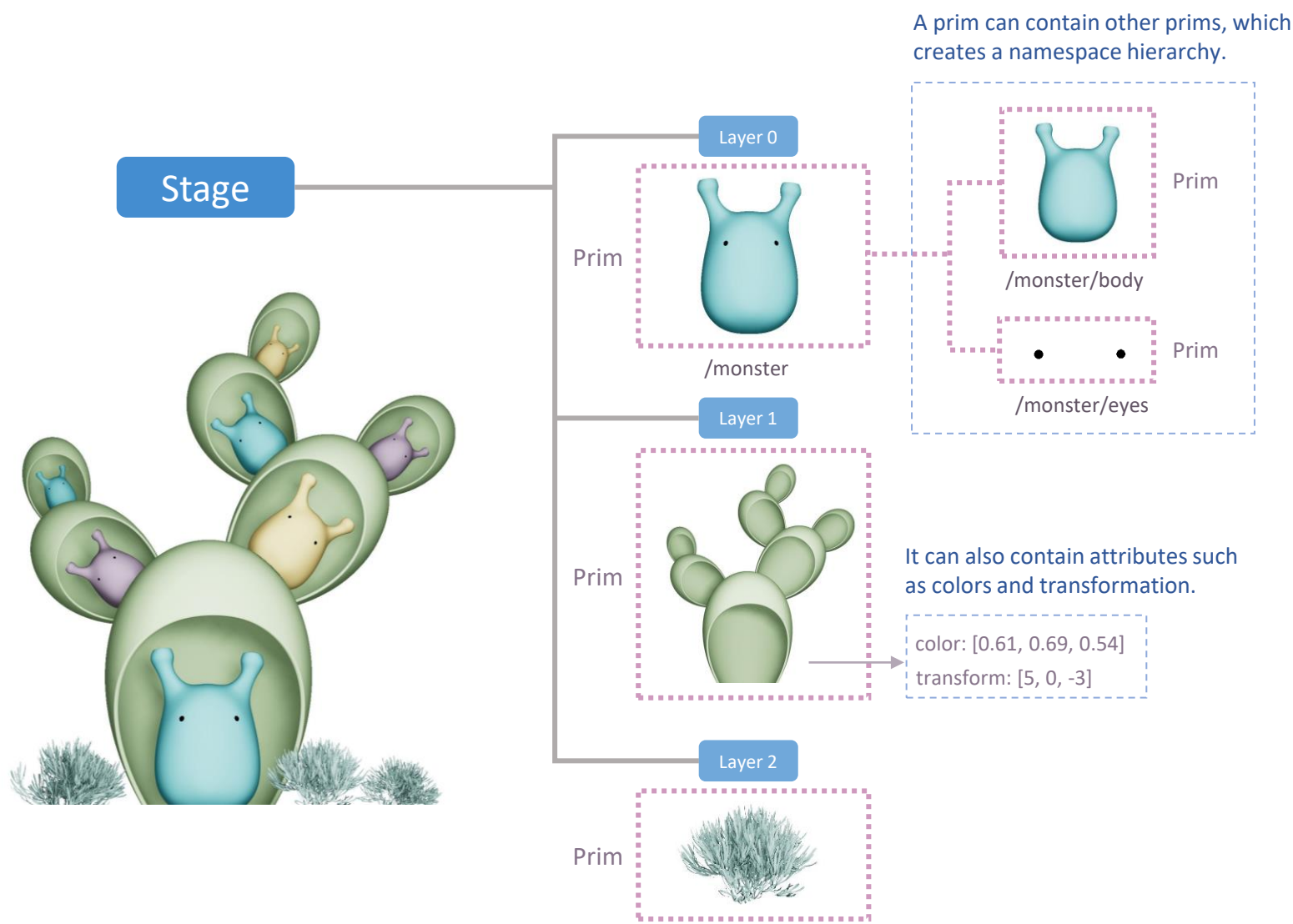
USD stage: a 3D scene composed of USD layer(s)



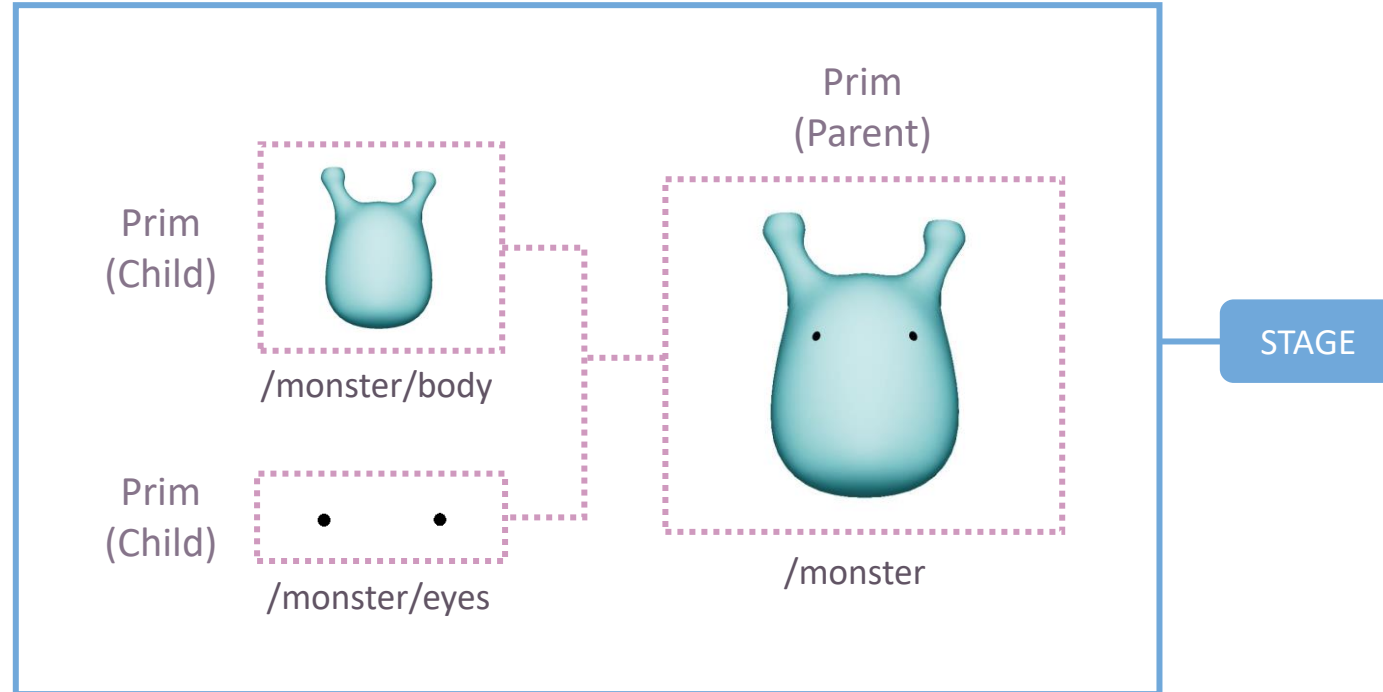
An example of a USD stage

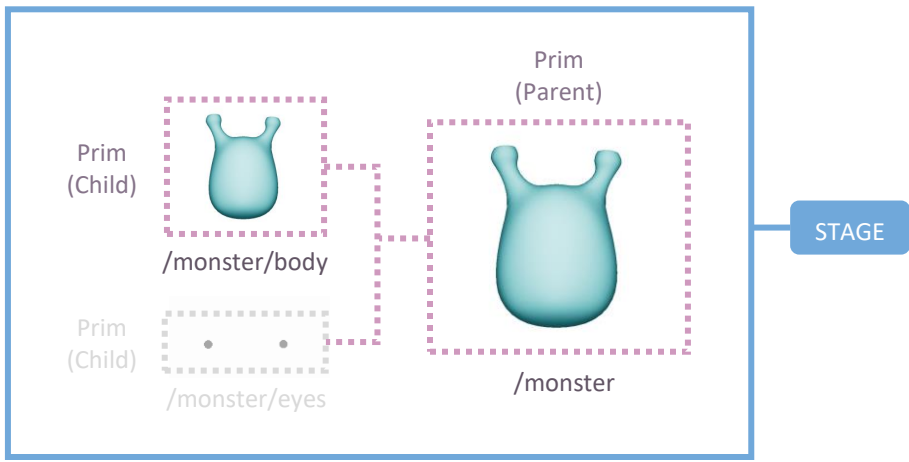
USD: Universal Scene Description

The basic container for a USD object is called a **Prim**.



Creating a Stage





create_usd_stage
Type: create_usd_stage

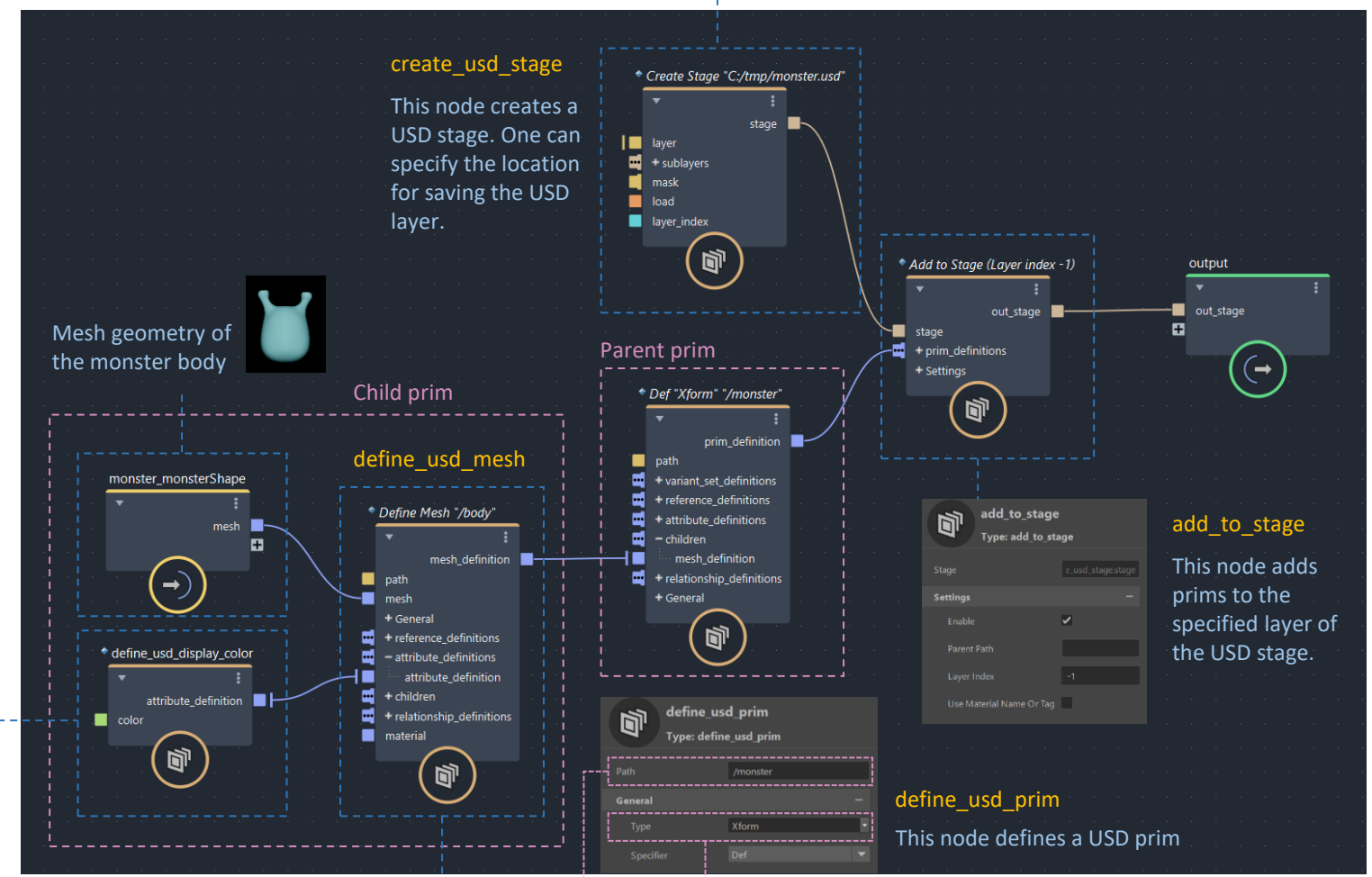
Layer: C:/tmp/monster.usd

Load: LoadAll

Layer Index: -1

Specify the location where you want to save the USD file

Layer index. -1 indicates the root layer



define_usd_display_color
Type: define_usd_display_color

Color: 0.1823 0.4518 0.5431

Attributes, such as color and transformation, can be added to a prim

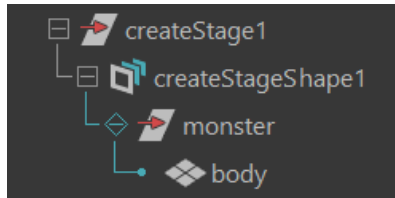
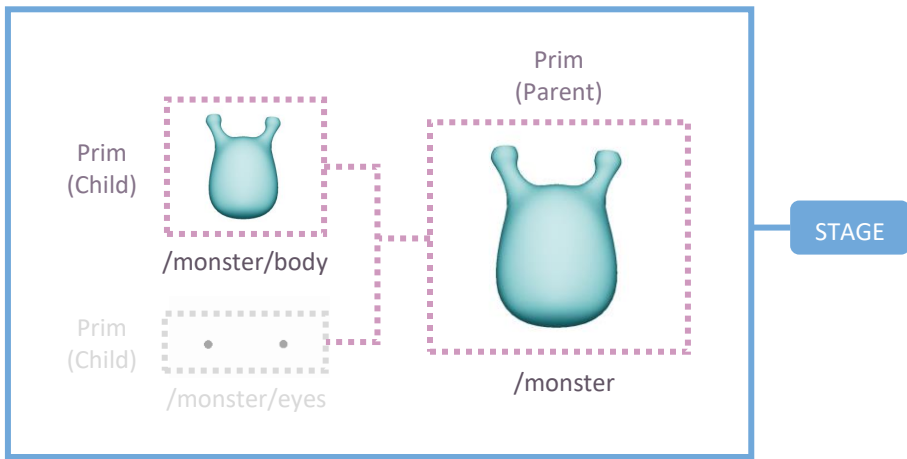
define_usd_mesh
Type: define_usd_mesh

Path: /body

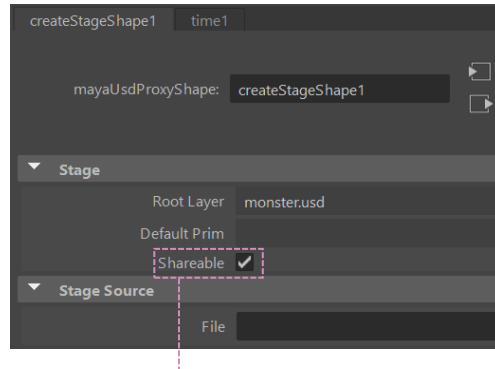
Mesh: createStageShape.mesh

Path of the prim

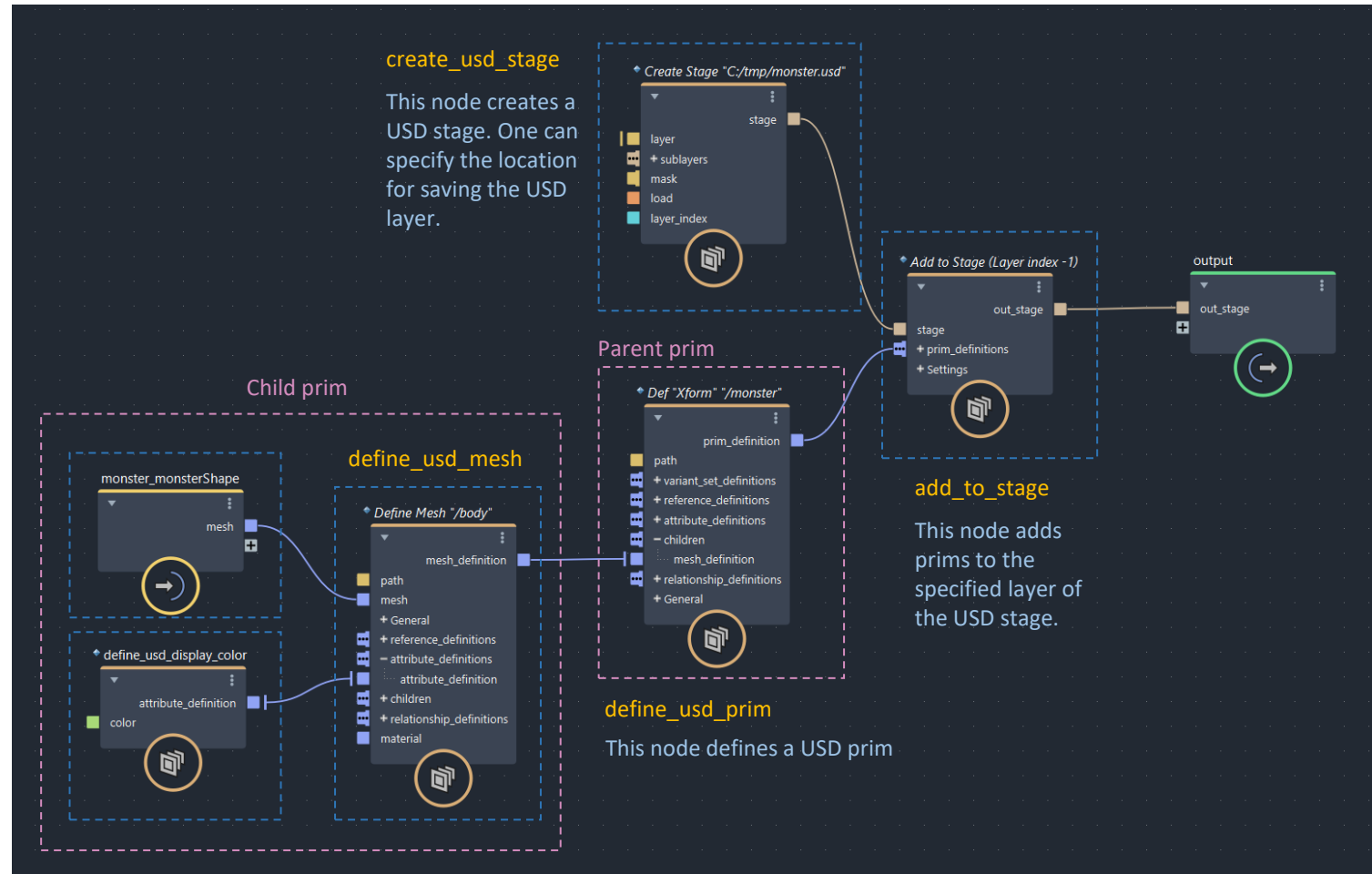
Type of the prim:
Xform acts as a container for other prims
Other types of prims include Scope, Mesh, Capsule, Cone, Cube, Sphere, etc.

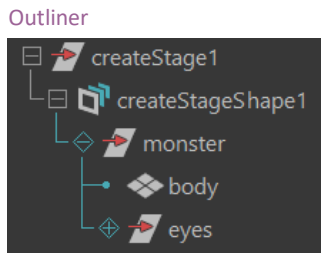
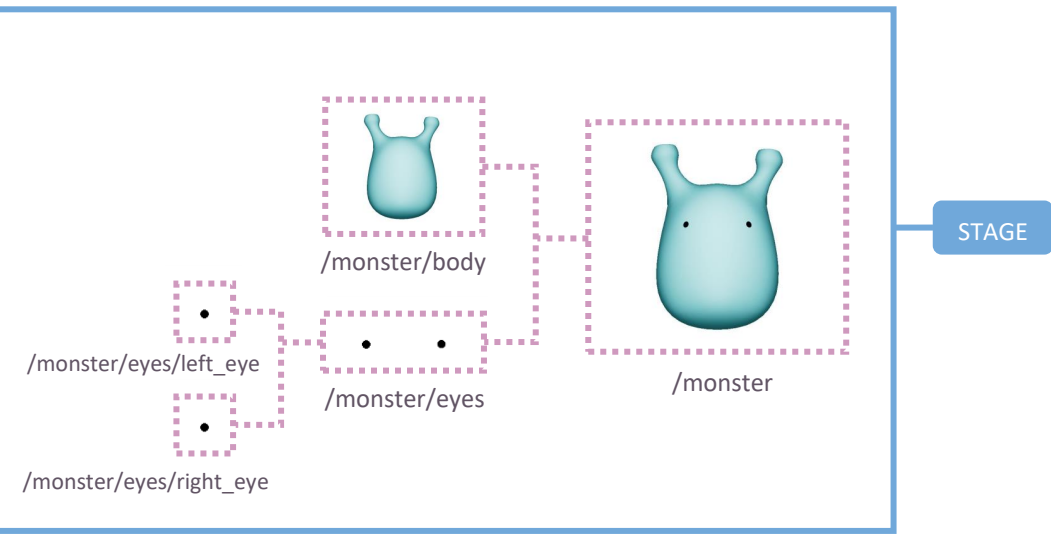


The hierarchy of the stage can be viewed in the outliner.



Make sure "Shareable" is turned on for the Maya USD proxy shape. This allows one to directly access the stage in the USD layer editor and set the target layer.





Transform attribute for the left eye

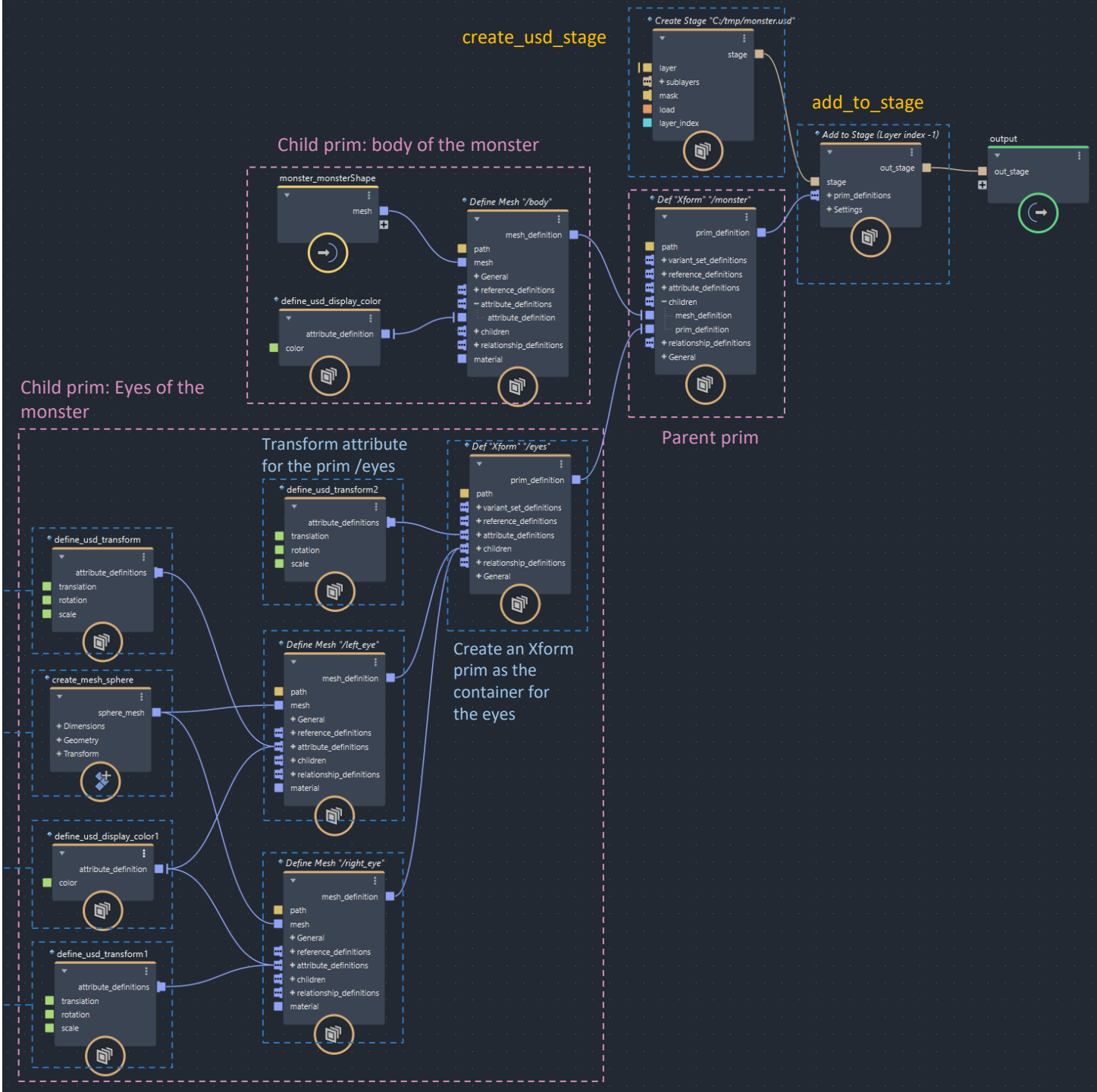
define_usd_transform			
Type: define_usd_transform			
Translation	-0.5955	2.126	0.7578
Rotation	0	0	0
Scale	0.1	0.1	0.1

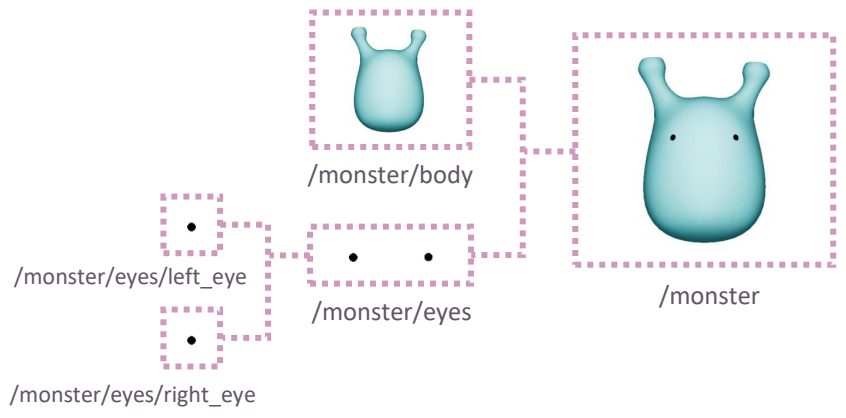
Mesh sphere for creating the eyes

define_usd_display_color1			
Type: define_usd_display_color			
Color	0	0	0

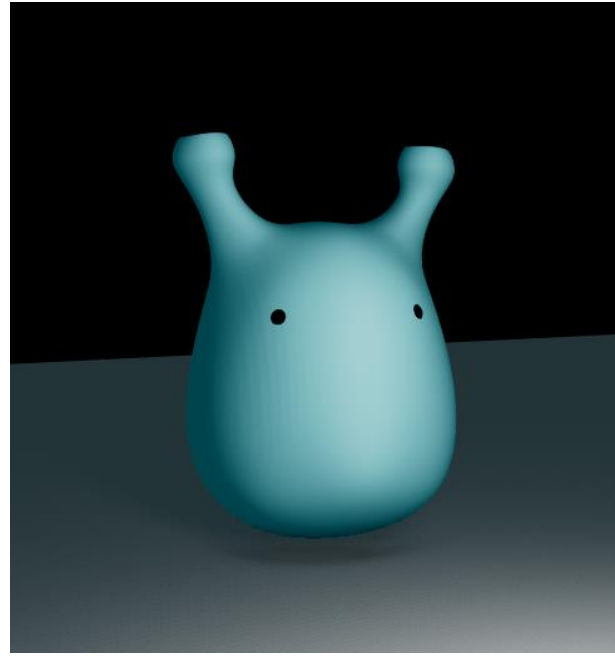
Transform attribute for the right eye

define_usd_transform1			
Type: define_usd_transform			
Translation	0.5955	2.126	0.7578
Rotation	0	0	0
Scale	0.1	0.1	0.1

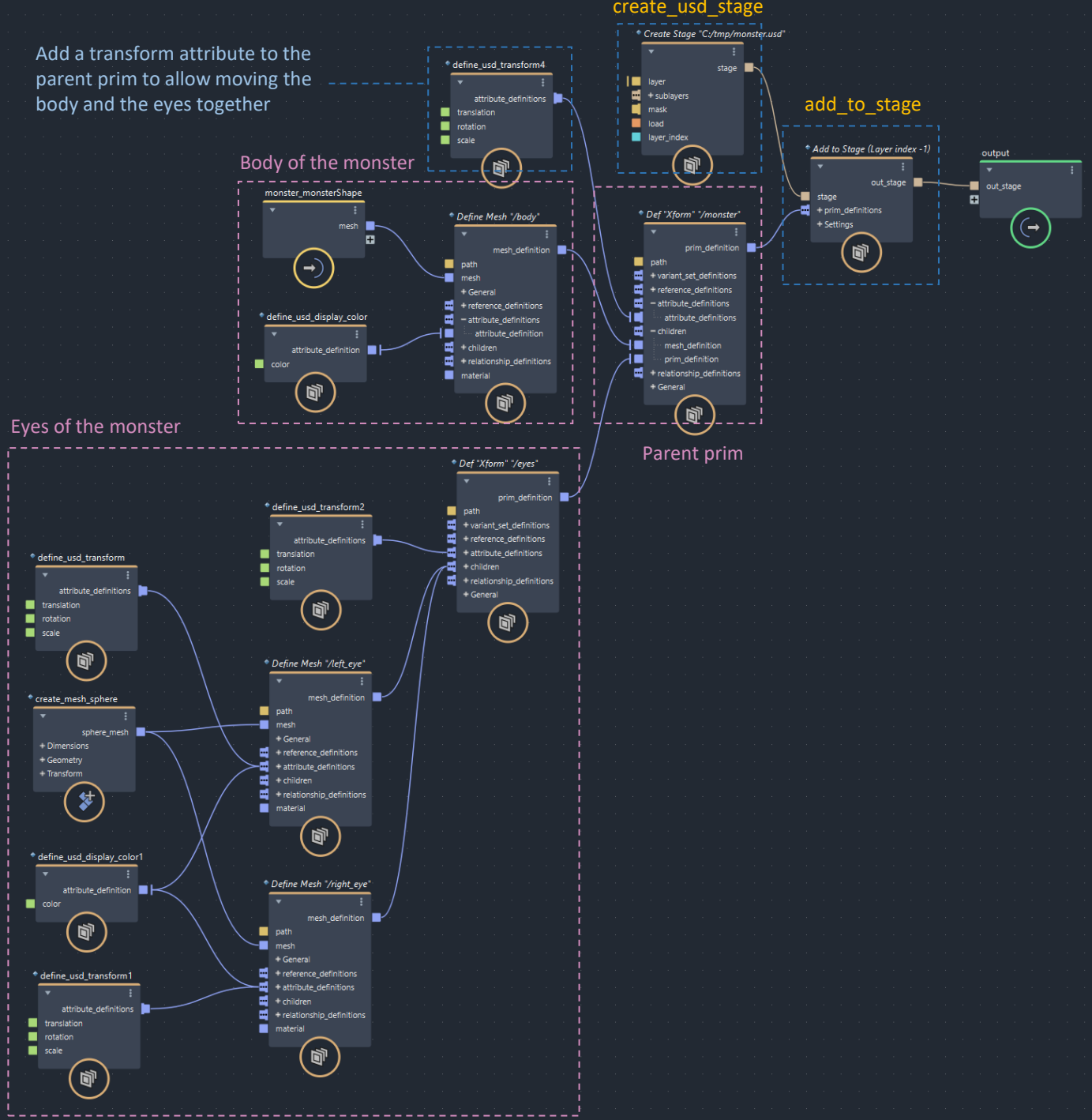




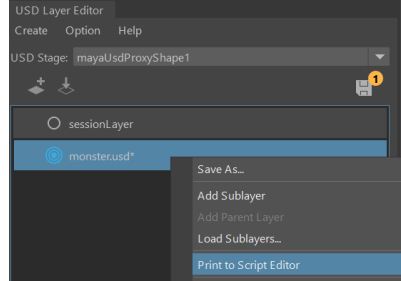
STAGE



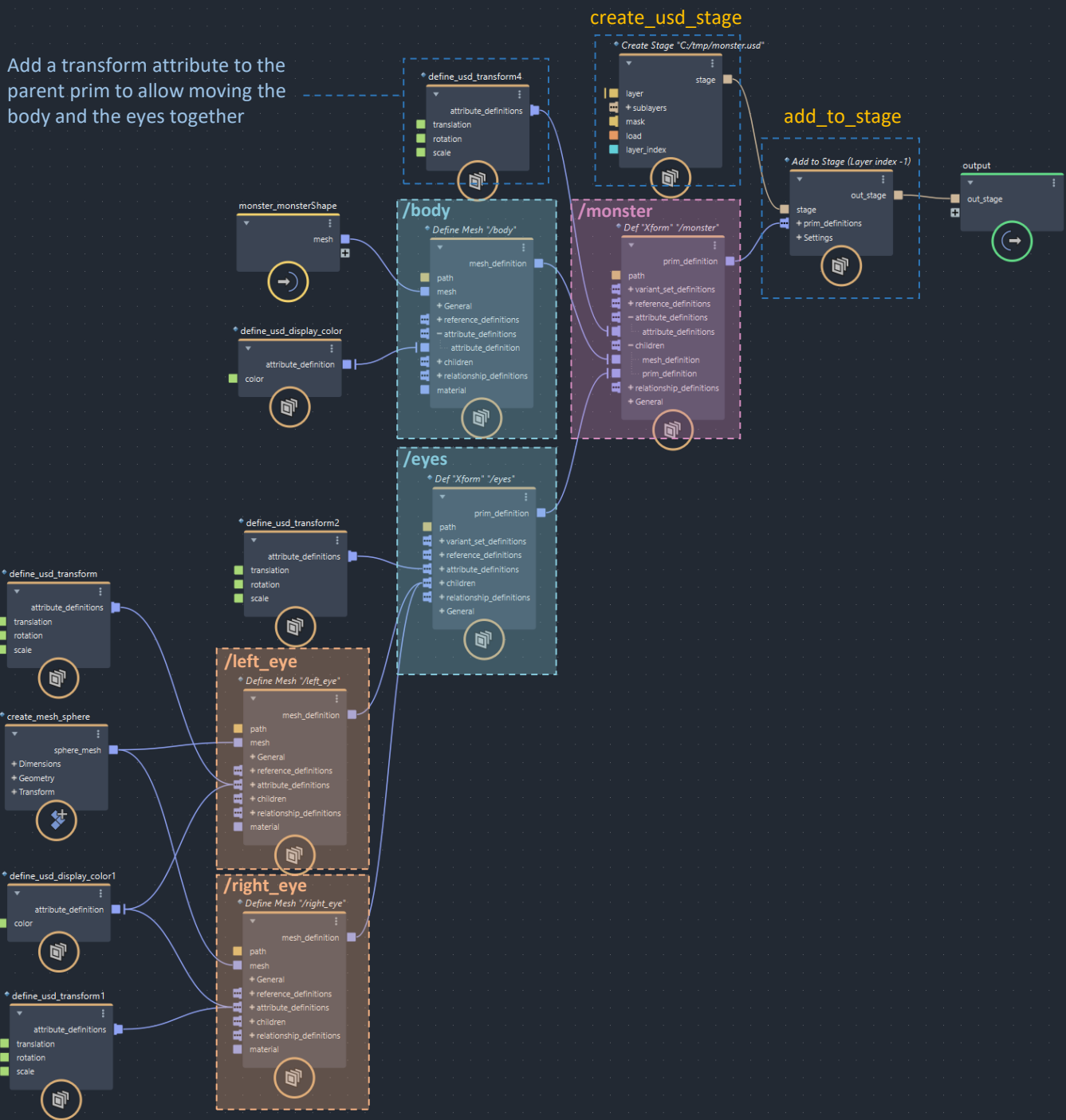
Add a transform attribute to the parent prim to allow moving the body and the eyes together



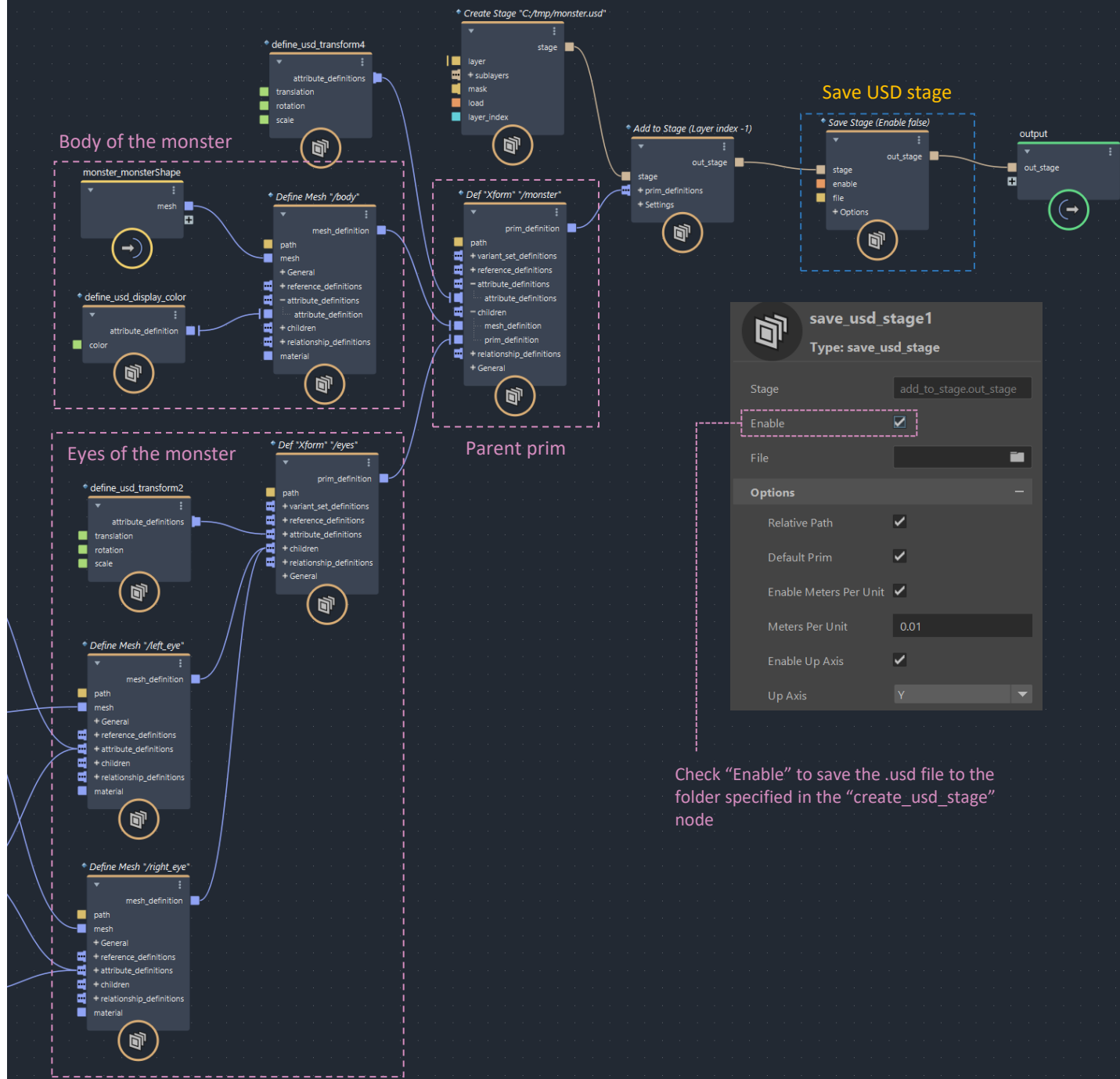
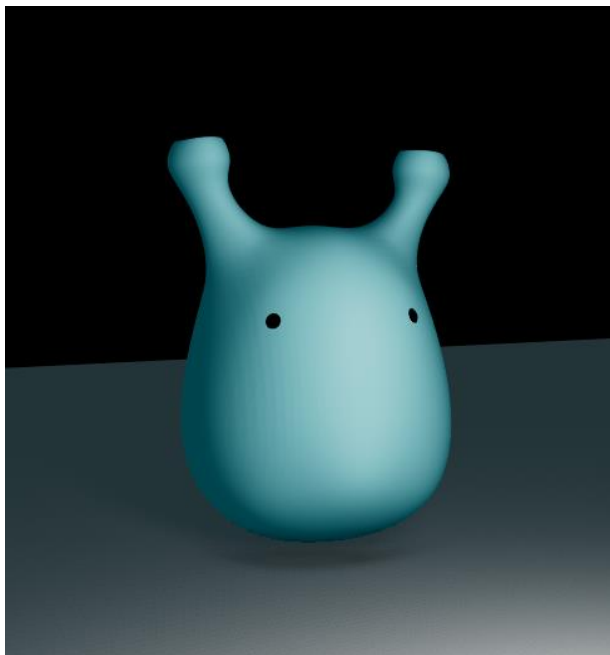
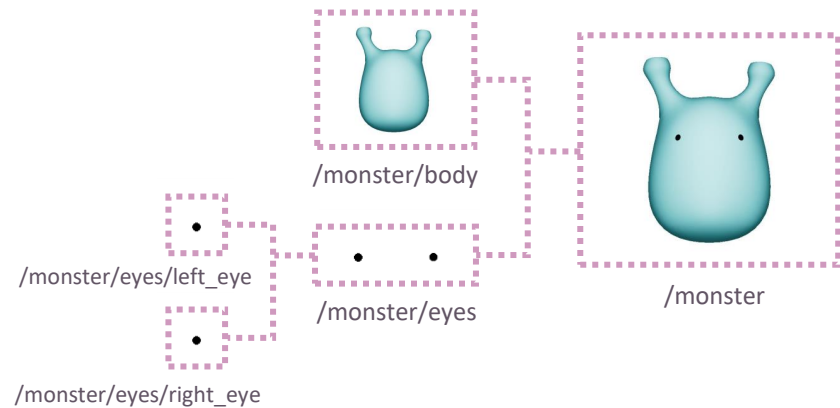
Print to script editor to view the commands

 Script Editor

Add a transform attribute to the parent prim to allow moving the body and the eyes together



monster.usd



Variant Sets



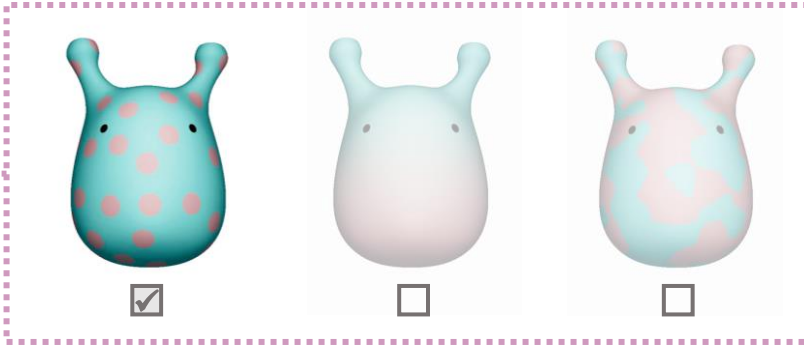
/monster/body/polka_dot



/monster/body/gradient



/monster/body/noise



/monster/body



/monster/eyes



/monster/flowers/dandelion



/monster/flowers/tulip



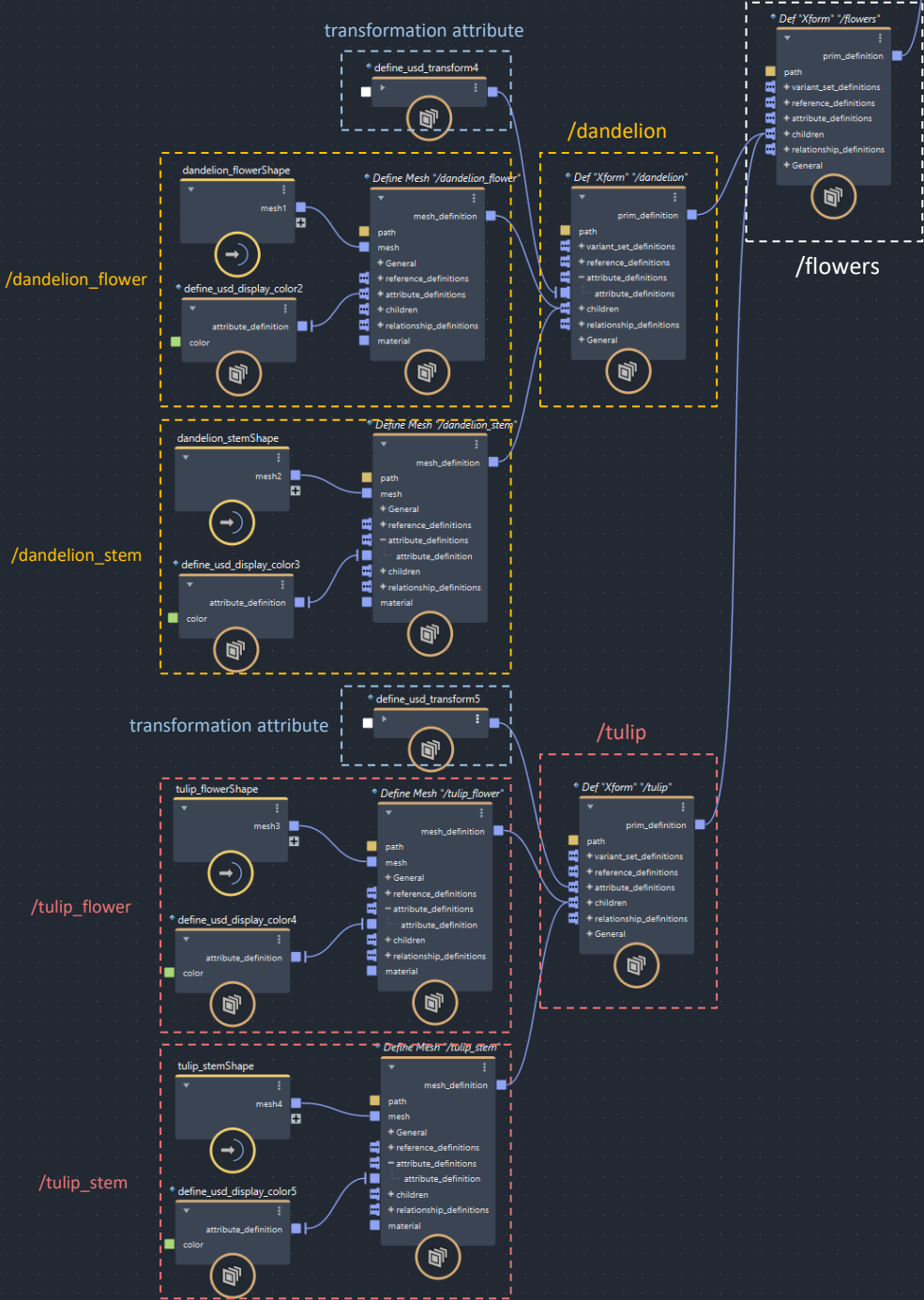
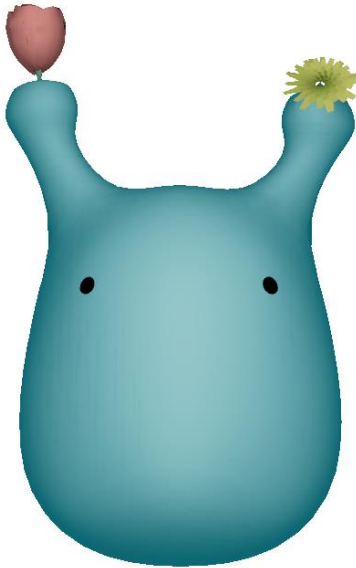
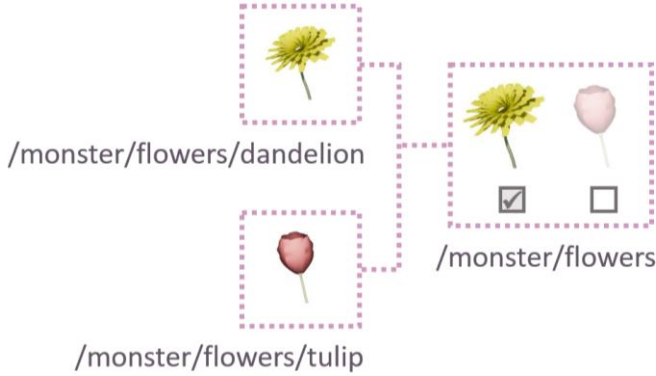
/monster/flowers



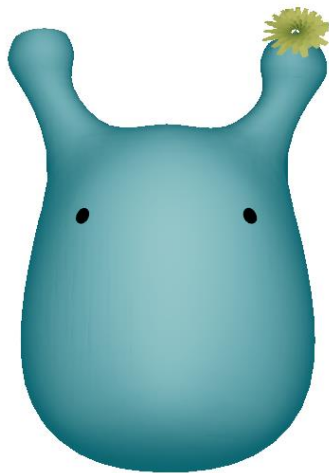
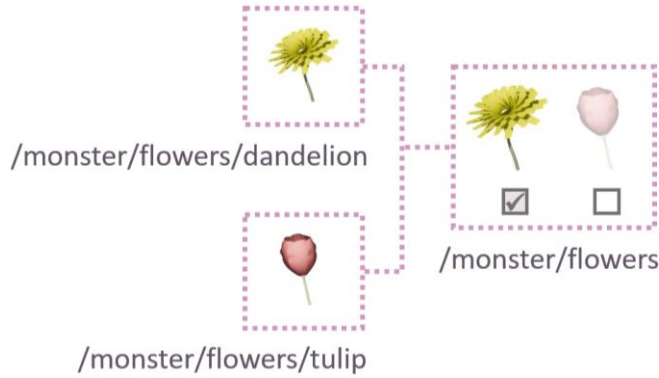
/monster

monster.usd

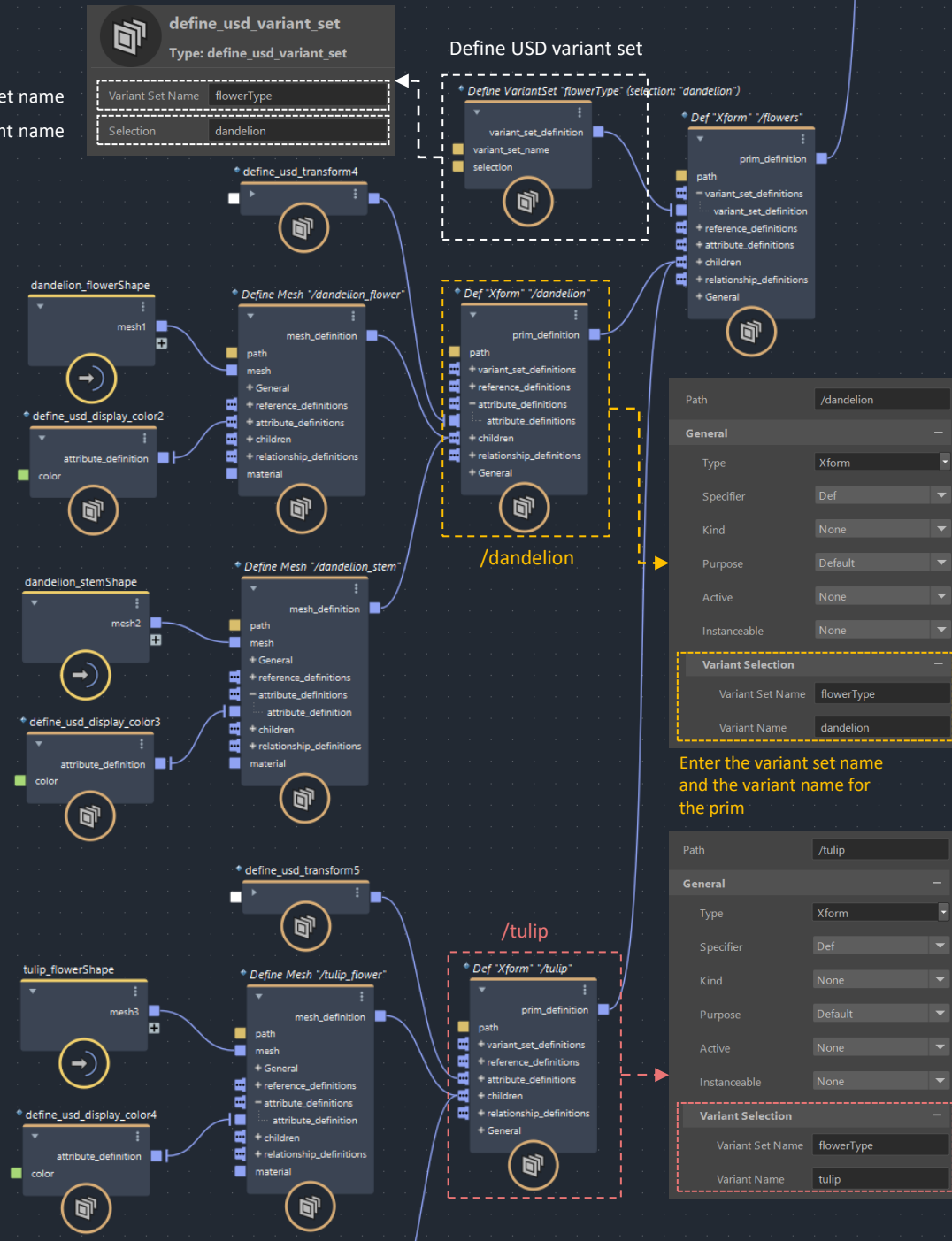
Variant Set 1: Flowers



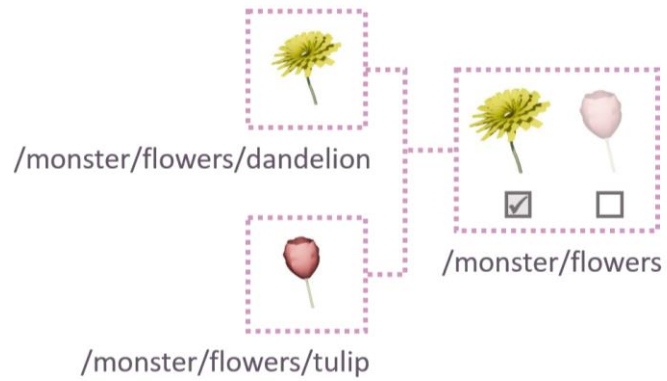
Variant Set 1: Flowers



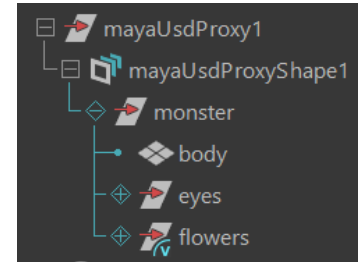
Variant set name
Default variant name



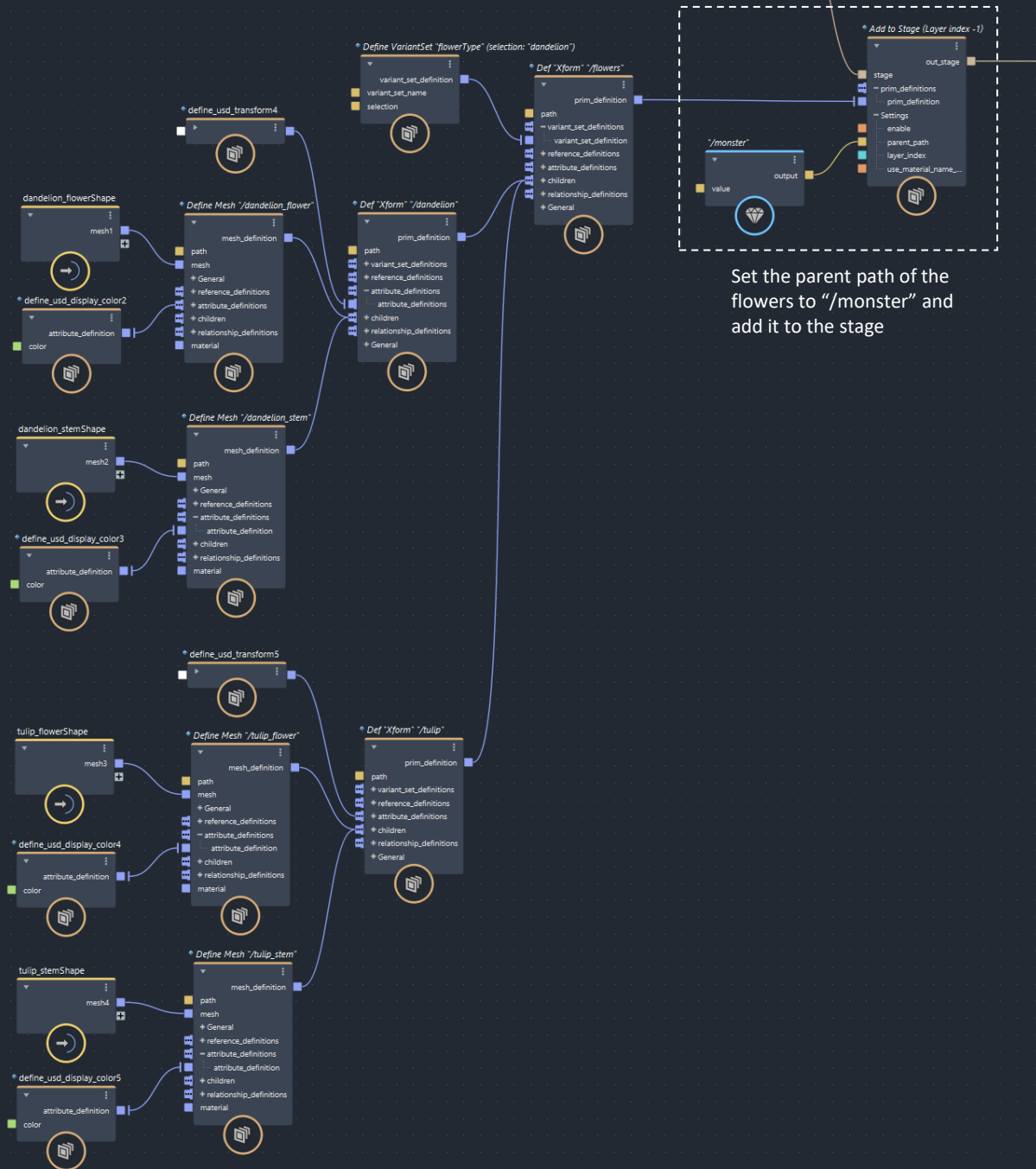
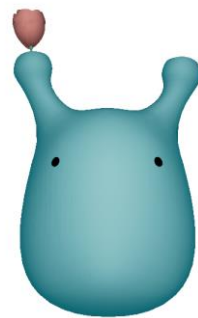
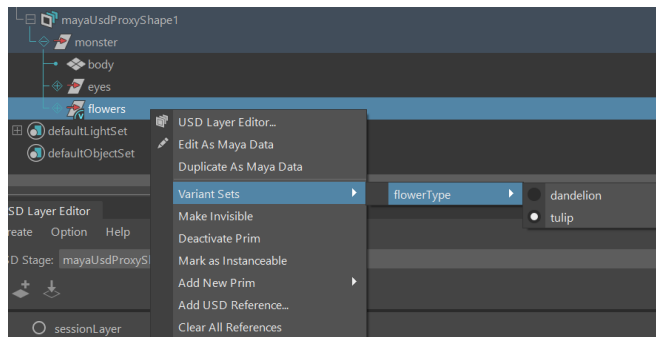
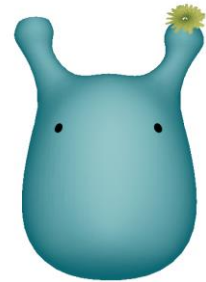
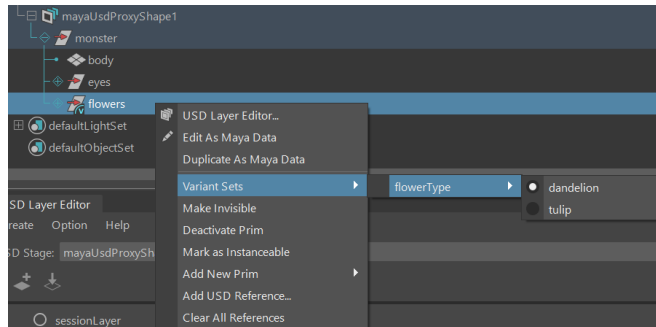
Variant Set 1: Flowers



Outliner



The letter "V" indicates that the prim is a variant set



Variant Set 1: Flowers

Script Editor

```
def xform "flower" {
    variants = {
        string flowerType = "dandelion"
    }
    prepend variantSets = "flowerType"
}

over "dandelion" {
    variants = {
        string flowerType = "dandelion"
    }
    over "dandelion_flower" {
        variants = {
            string flowerType = "dandelion"
        }
    }
    over "dandelion_stem" {
        variants = {
            string flowerType = "dandelion"
        }
    }
}

variantSet "flowerType" = {
    "dandelion" {
        def Xform "dandelion"

        float3 xformOp:rotateXYZ = (0, 10, 0)
        float3 xformOp:scale = (1.5, 1.5, 1.5)
        float3 xformOp:translate = (-0.5460, 2.15, -0.325)
        uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ", "xformOp:scale"]

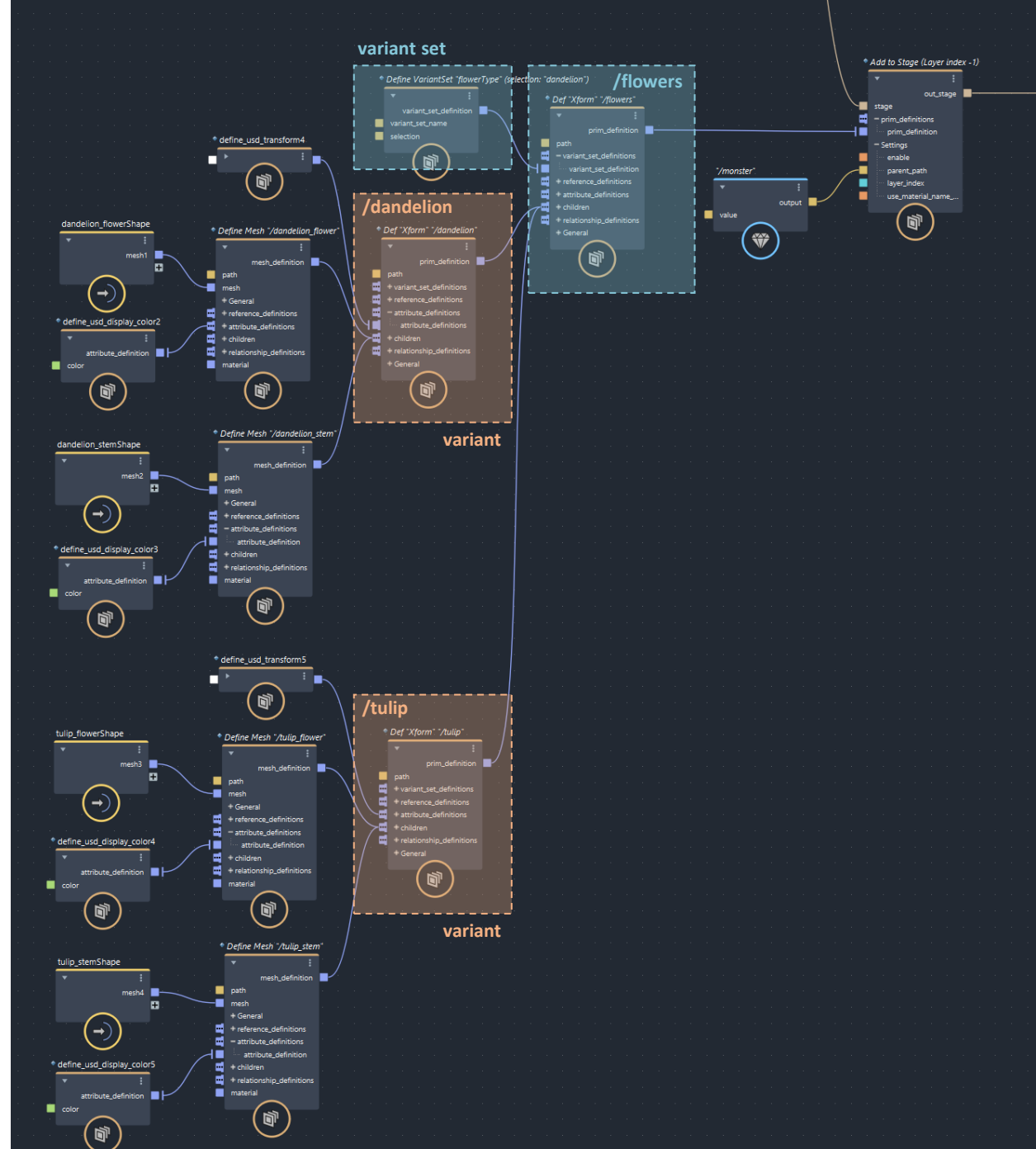
        def Mesh "dandelion_flower"
        {
            int[] faceVertexCounts = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
            int[] faceVertexIndices = [0, 3, 2, 1, 3, 0, 4, 7, 6, 5, 7, 4, 8, 11, 10, 9, 11, 8, 12, 1]
            point3f[] points = [(1.019178, 0.937426, 0.524912), (1.005502, 0.979946, 0.489701), (1.06
            color3f[] primvars:displayColor = [(0.8428, 0.7946, 0.0589)] {
                interpolation = "constant"
            }
            normal3f[] primvars:normals = [(0.23598611, -0.5736373, -0.78437936), (0.2333741, -0.5745
                interpolation = "faceVarying"
            )
            texCoord2f[] primvars:st = [(0, 0), (1, 0.25), (0, 0.25), (1, 0), (0, 0), (1, 0.25), (0,
                interpolation = "faceVarying"
            )
            int[] primvars:st:indices = [0, 1, 2, 3, 1, 0, 4, 5, 6, 7, 5, 4, 8, 9, 10, 11, 9, 8, 12,
                uniform token subdivisionScheme = "none"
            )
        }

        def Mesh "dandelion_stem"
        {
            int[] faceVertexCounts = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
            int[] faceVertexIndices = [0, 4, 3, 1, 4, 0, 1, 5, 4, 2, 5, 1, 2, 3, 5, 0, 3, 2, 3, 7, 6]
            point3f[] points = [(1.047543, 0.724175, 0.295977), (1.031263, 0.720765, 0.295099), (1.04
            color3f[] primvars:displayColor = [(0.1835, 0.3415, 0.1791)] {
                interpolation = "constant"
            }
            normal3f[] primvars:normals = [(-0.17706561, 0.65975326, -0.7303241), (-0.17706564, 0.659
                interpolation = "faceVarying"
            )
            texCoord2f[] primvars:st = [(0, 0.55), (0.333333, 0.6), (0, 0.6), (0.333333, 0.55), (0.66
                interpolation = "faceVarying"
            )
            int[] primvars:st:indices = [0, 1, 2, 3, 1, 0, 3, 4, 1, 5, 4, 3, 5, 6, 4, 7, 6, 5, 2, 8,
                uniform token subdivisionScheme = "none"
            )
        }
    }
}

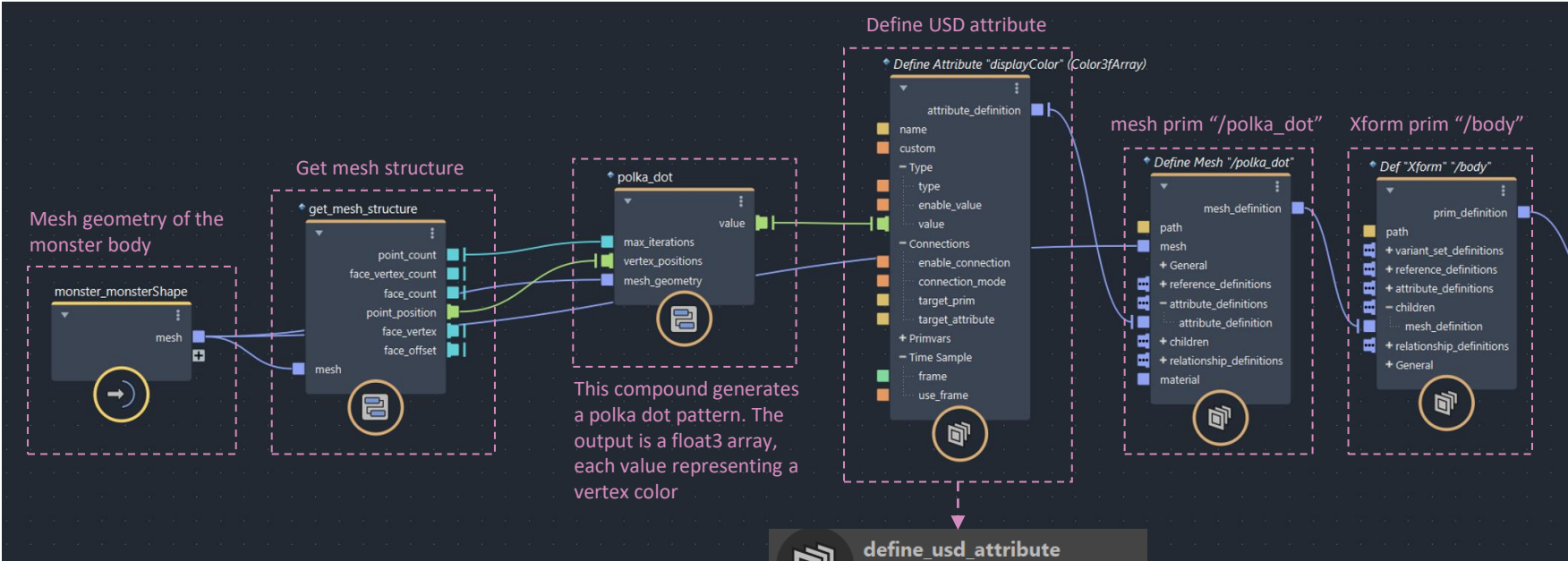
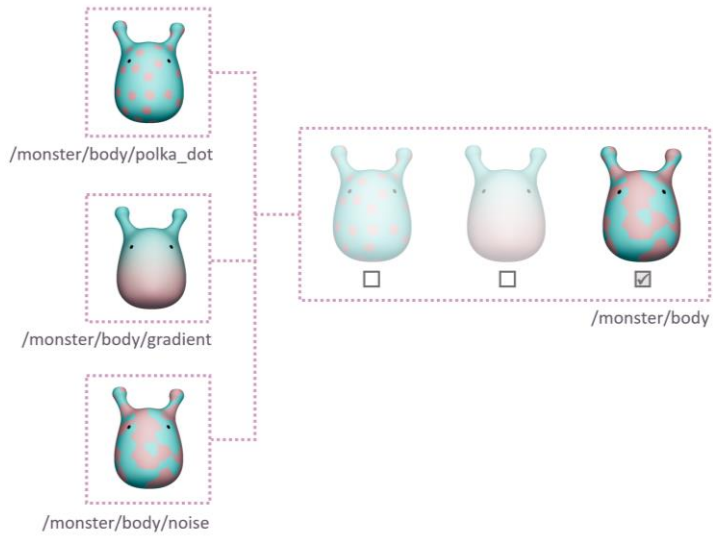
"tulip" {
    def Xform "tulip"

    float3 xformOp:rotateXYZ = (0, 0, 10)
    float3 xformOp:scale = (1.3, 1.3, 1.3)
    float3 xformOp:translate = (-0.8914, 1.615, -0.4081)
    uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ", "xformOp:scale"]

    def Mesh "tulip_flower"
    {
        int[] faceVertexCounts = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
        int[] faceVertexIndices = [0, 6, 5, 1, 6, 0, 1, 7, 6, 2, 7, 1, 2, 8, 7, 3, 8, 2, 3, 5, 8]
        point3f[] points = [(0.122411, 1.595252, 0.561128), (0.117034, 1.589891, 0.550191), (0.14
        color3f[] primvars:displayColor = [(0.6448, 0.124, 0.0811)] {
            interpolation = "constant"
        }
        normal3f[] primvars:normals = [(0.23598611, -0.5736373, -0.78437936), (0.2333741, -0.5745
            interpolation = "faceVarying"
        )
        texCoord2f[] primvars:st = [(0, 0.55), (0.333333, 0.6), (0, 0.6), (0.333333, 0.55), (0.66
            interpolation = "faceVarying"
        )
        int[] primvars:st:indices = [0, 1, 2, 3, 1, 0, 3, 4, 1, 5, 4, 3, 5, 6, 4, 7, 6, 5, 2, 8,
            uniform token subdivisionScheme = "none"
        )
    }
}
```



Variant Set 2: Color Patterns



Set the attribute name to "displayColor"

Select the type "Color3fArray" and check "Enable Value"

Check "Enable Primvar" and set the Interpolation to "PrimVarVertex"

define_usd_attribute
Type: define_usd_attribute

Name:

Custom: ☐

Type:

Enable Value: ☒

Connections:

Enable Connection: ☐

Connection Mode:

Target Prim:

Target Attribute:

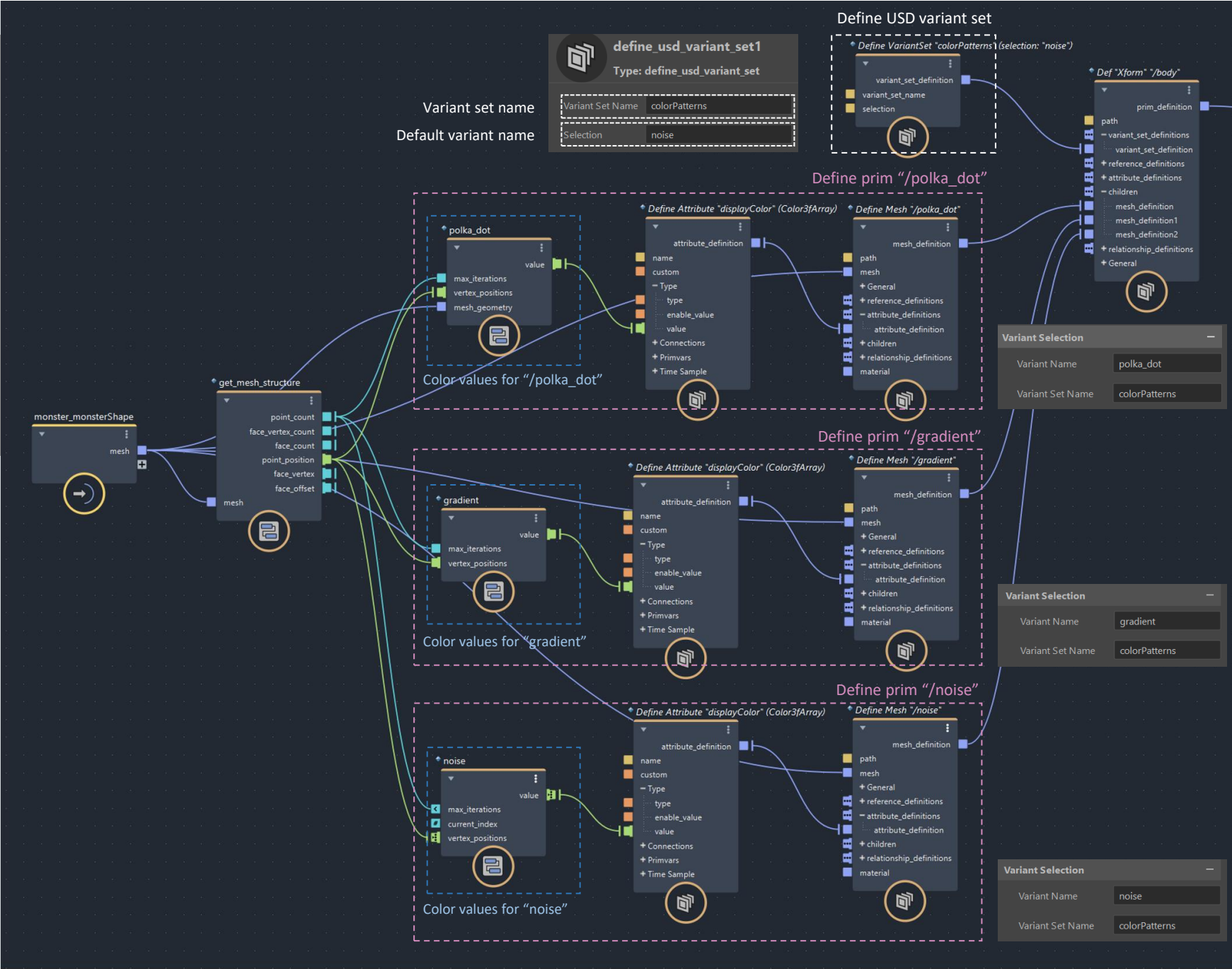
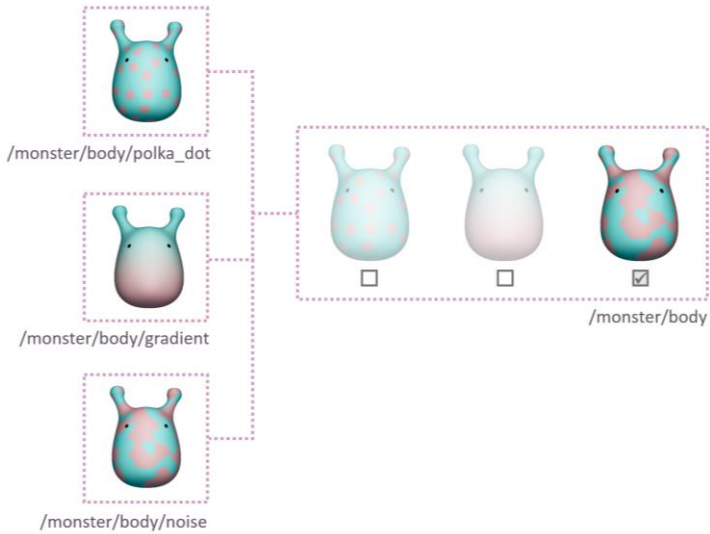
Primvars:

Enable Primvar: ☒

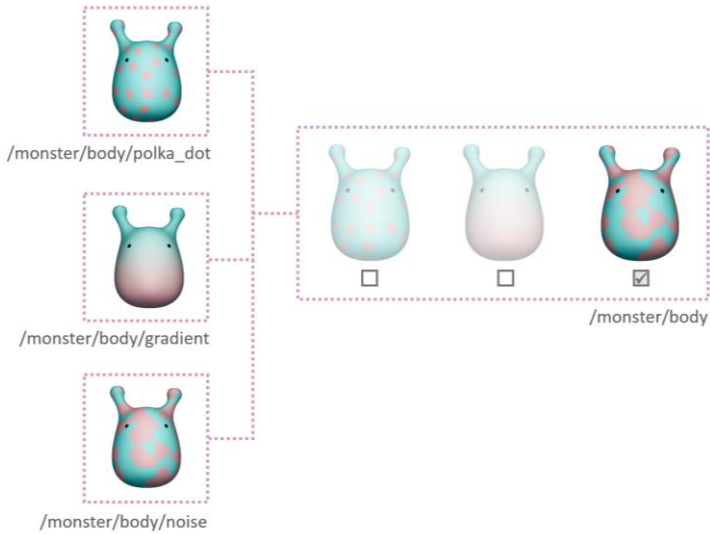
Interpolation:

Enabling Primvar allows one to associate an attribute with the geometric primitives, in this case, the vertices of the mesh. This way the value of the attribute can vary over the surface or volume.

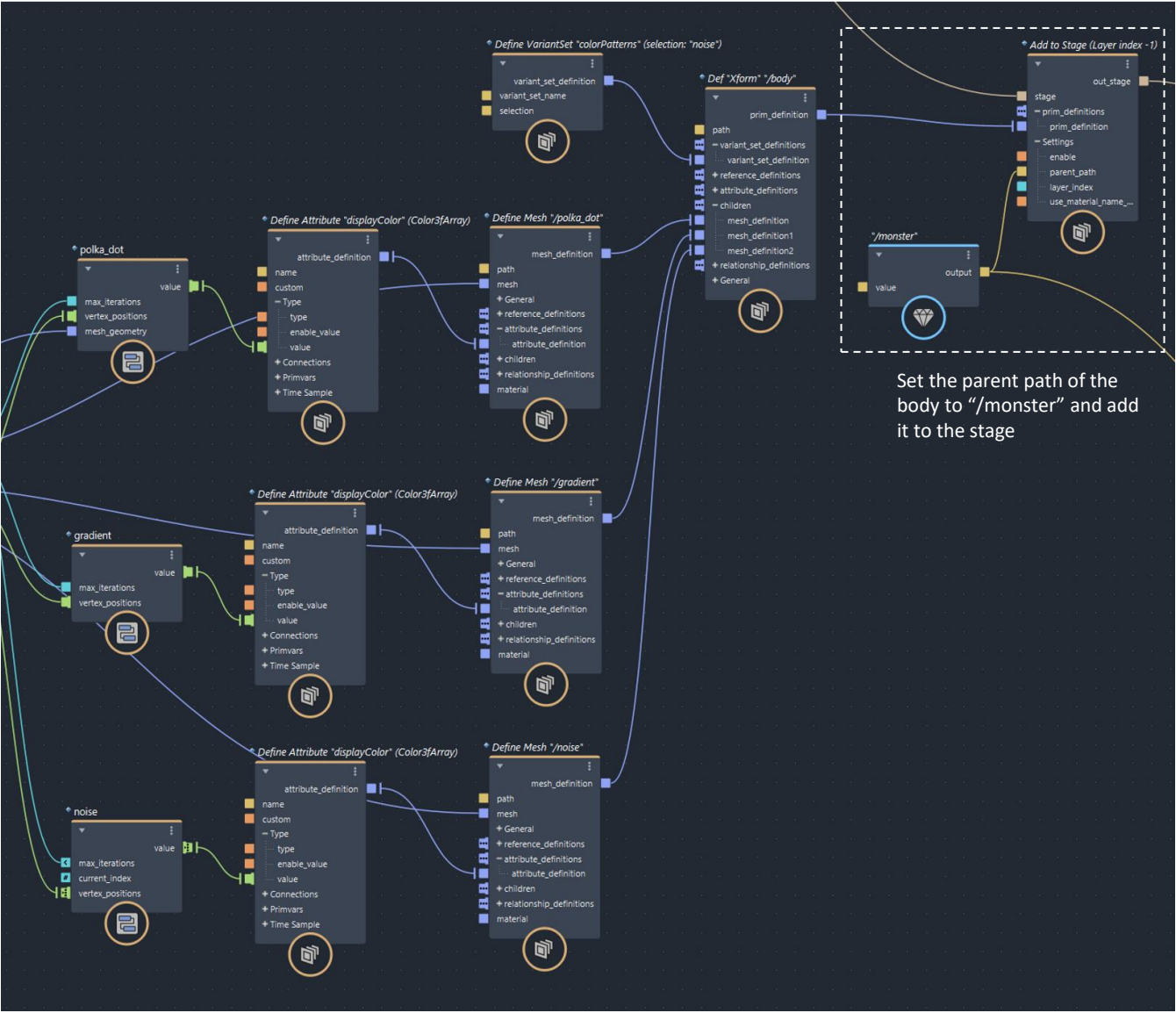
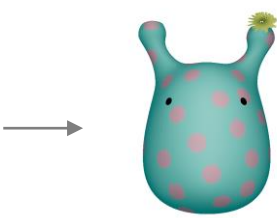
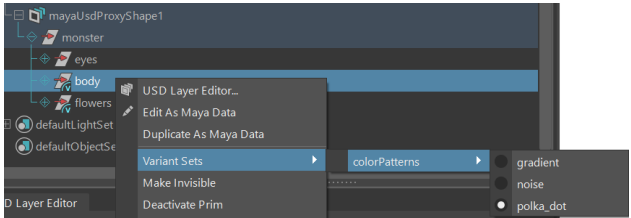
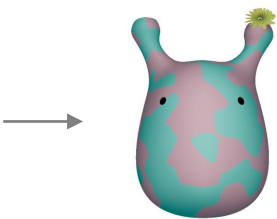
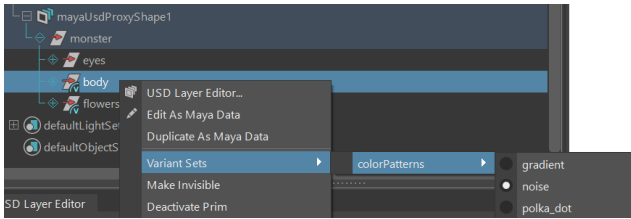
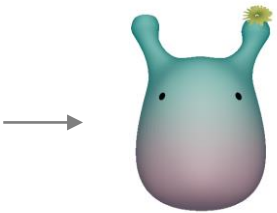
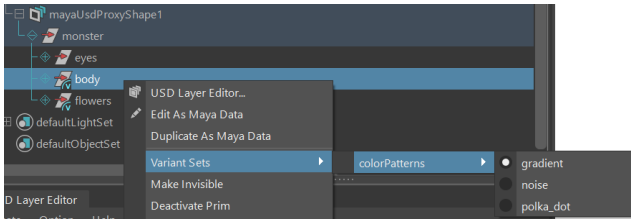
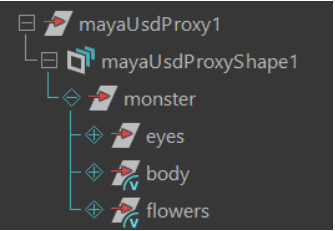
Variant Set 2: Color Patterns



Variant Set 2: Color Patterns

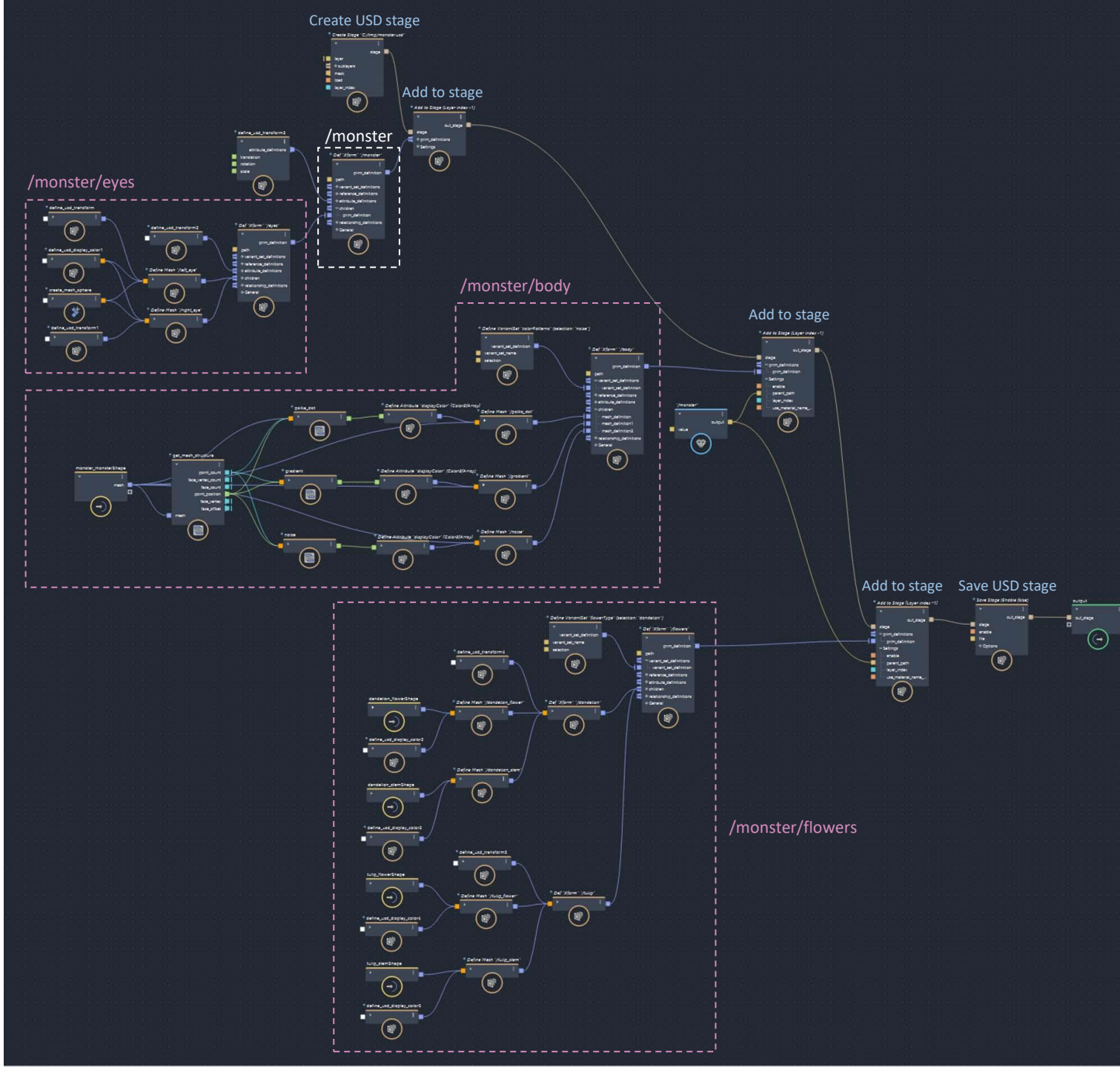
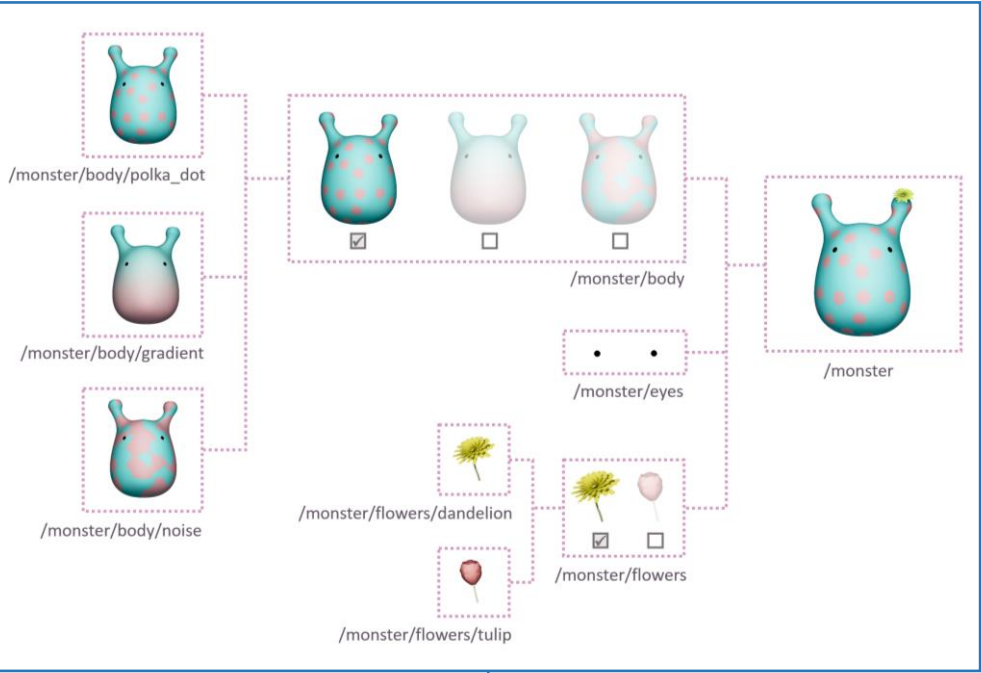


Outliner

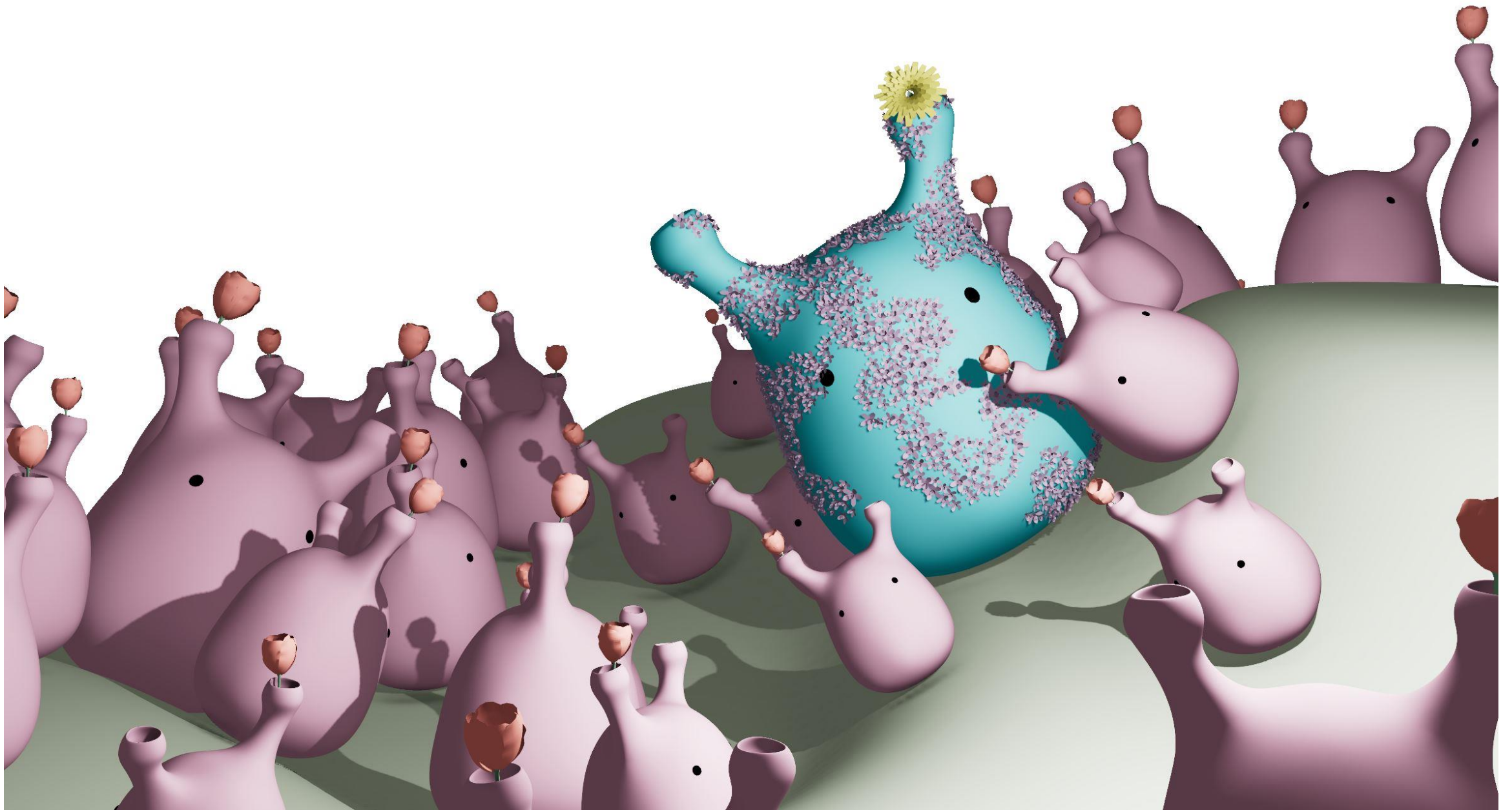


Set the parent path of the body to "/monster" and add it to the stage

Variant Set 2: Color Patterns



USD Instancing

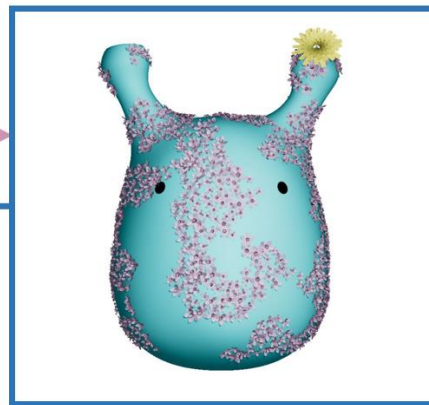


Root Layer:
monsterland.usd

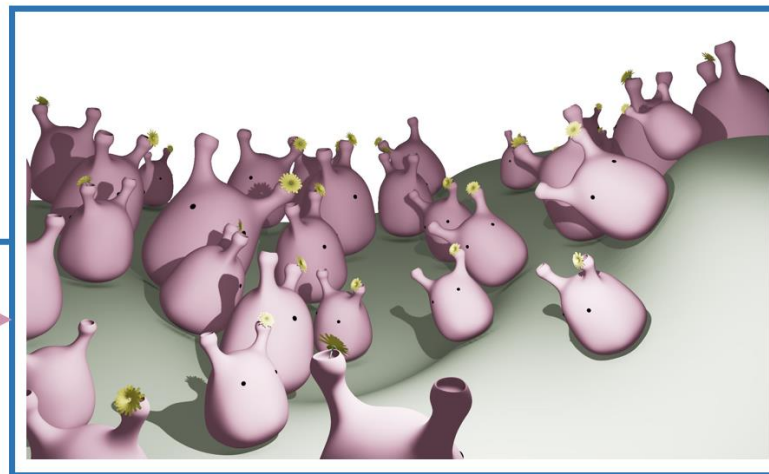
reference: monster.usd



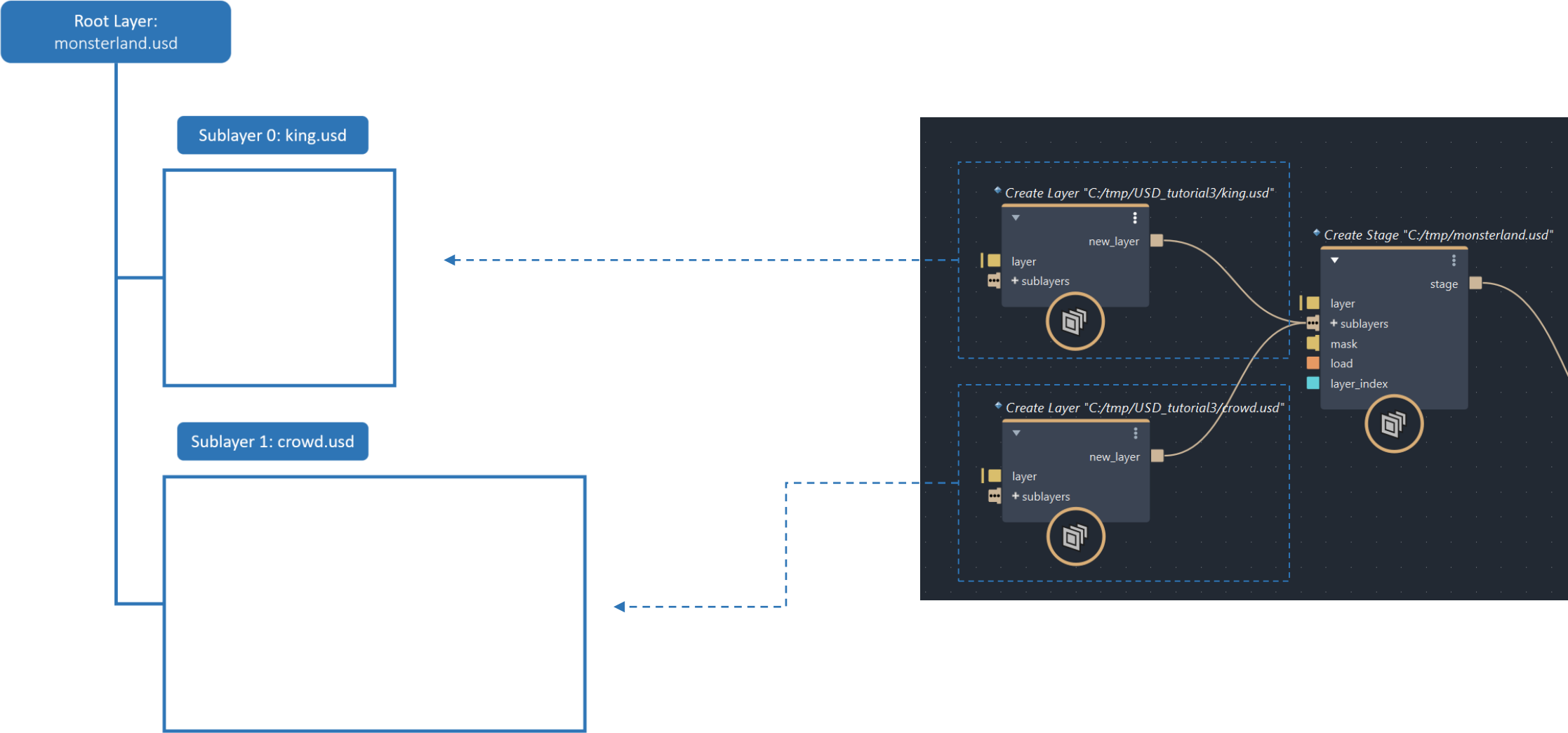
Sublayer 0: king.usd



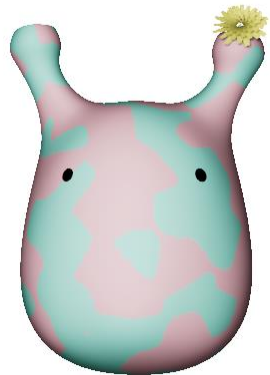
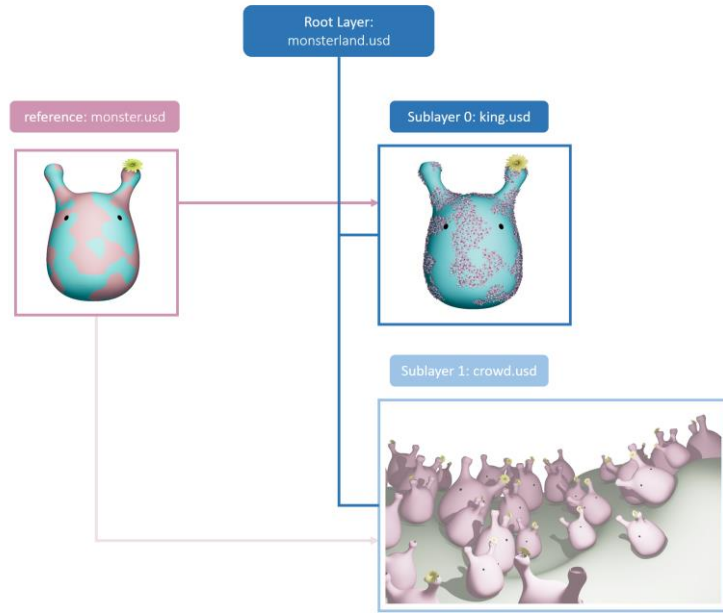
Sublayer 1: crowd.usd



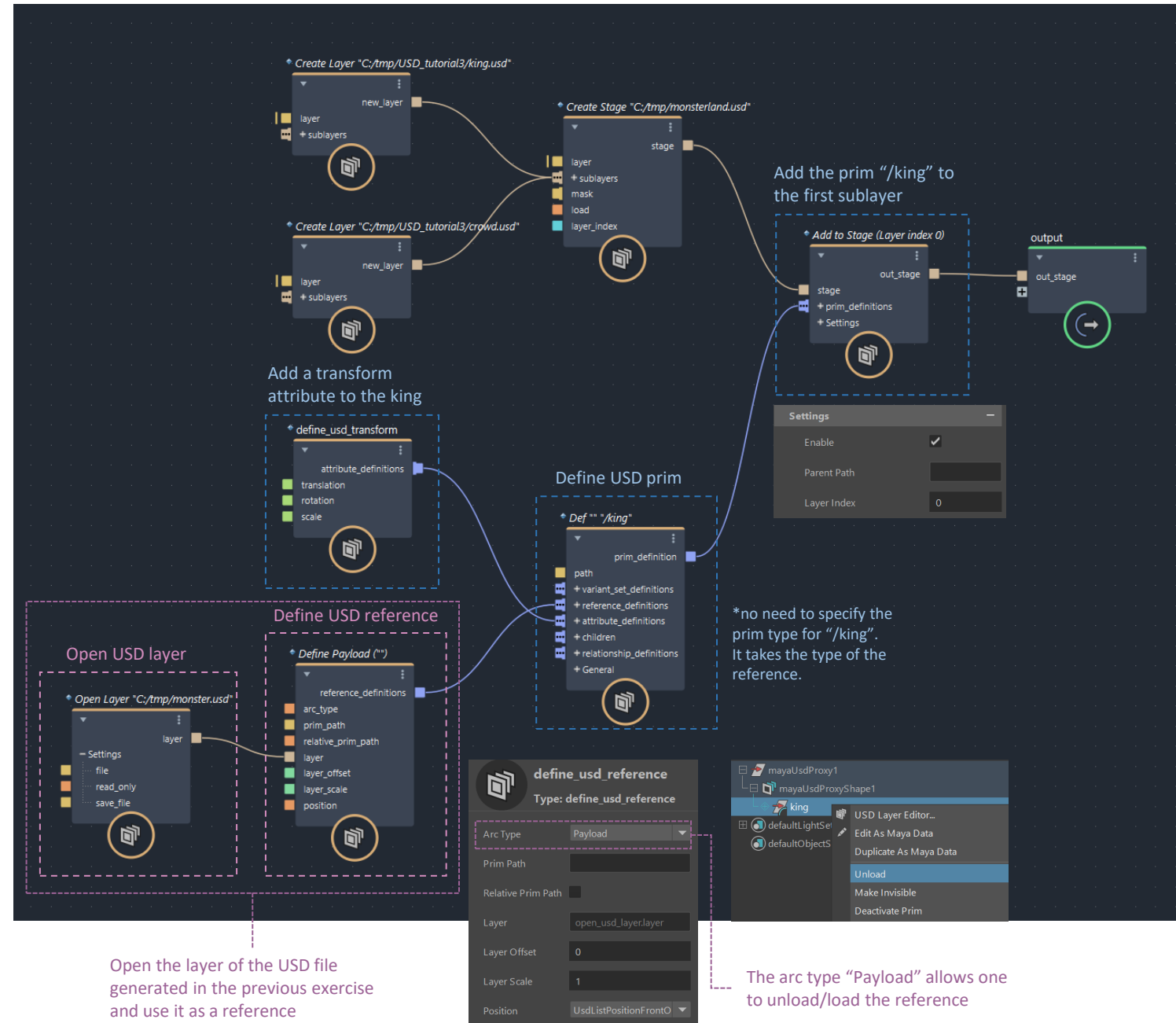
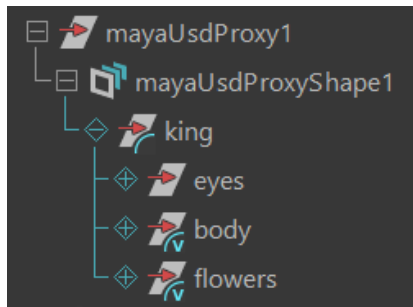
Create a stage with two sublayers



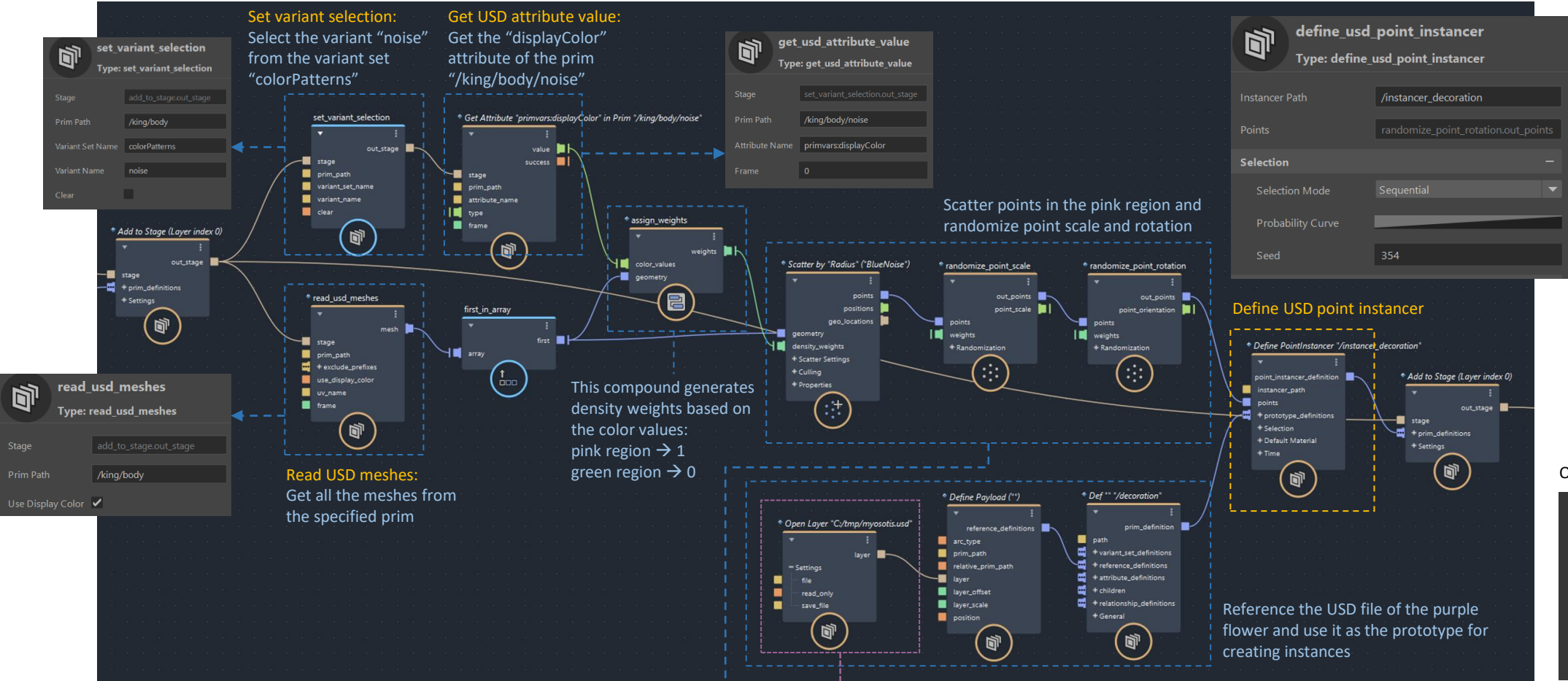
Create the king: define USD reference



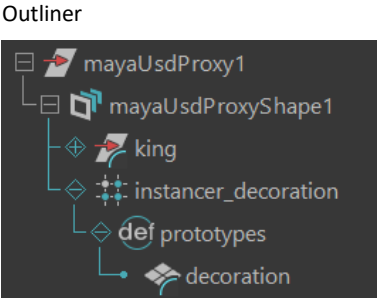
Outliner



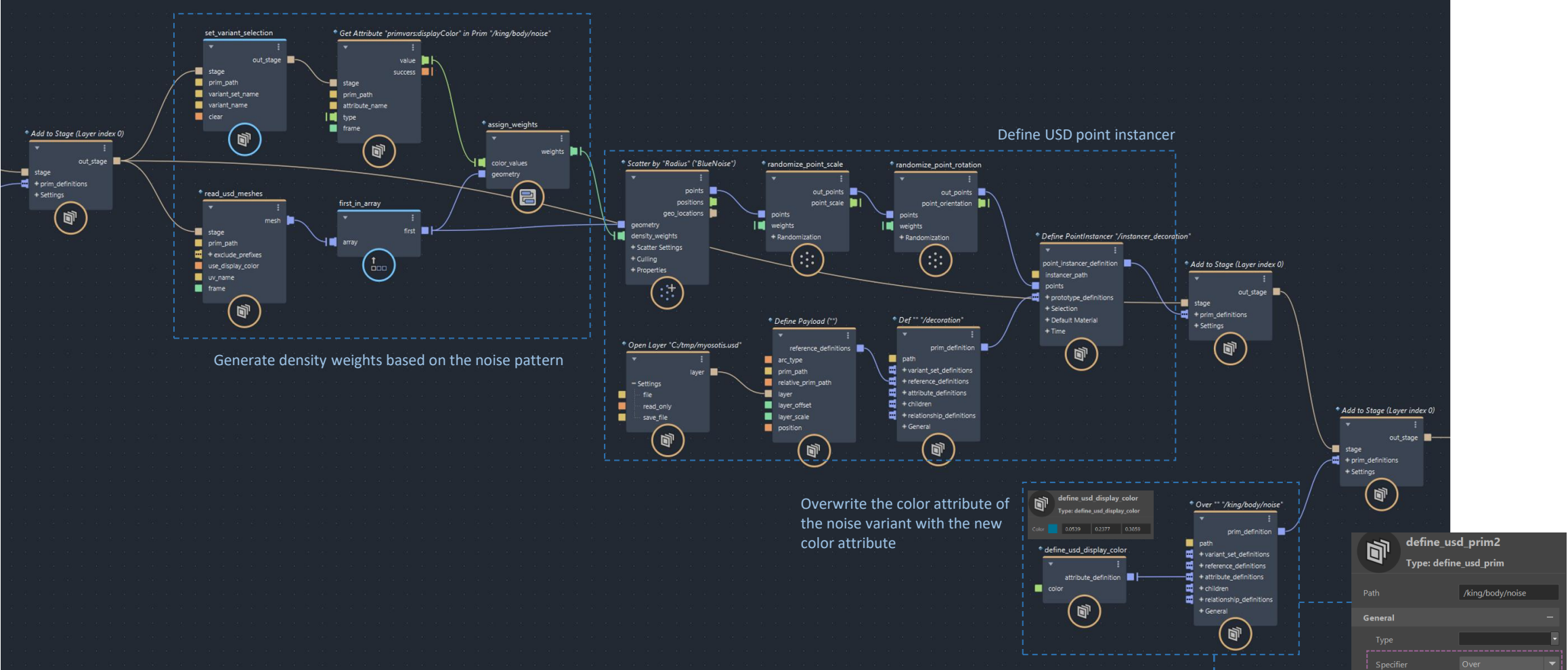
Create the king: define USD point instancer



"define_point_instancer" defines a USD point instancer prim based on a Bifrost points object. It creates instances when added to the stage

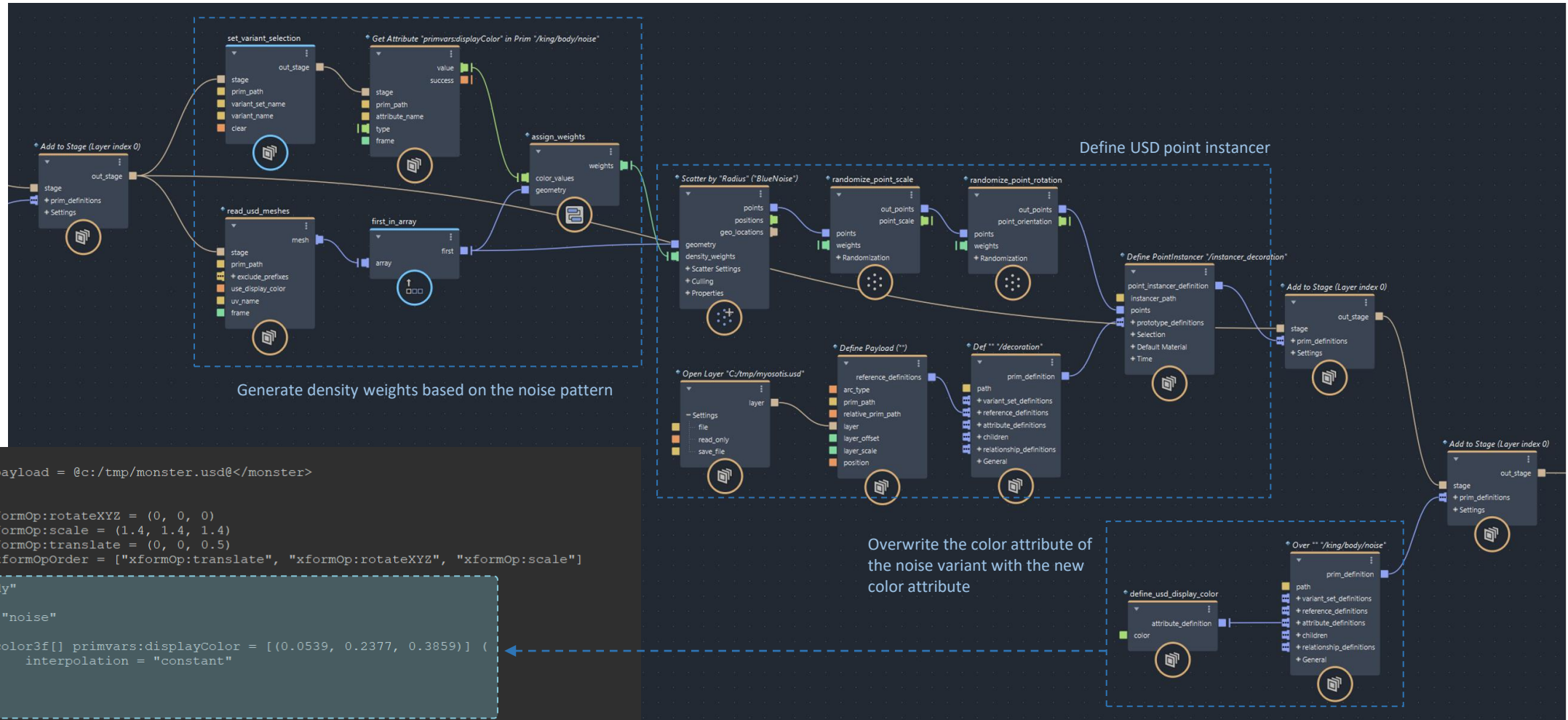


Create the king: define USD point instancer



If the specifier is set to "Over", the provided definitions will overwrite the existing ones of the specified prim

Create the king: define USD point instancer



Script Editor

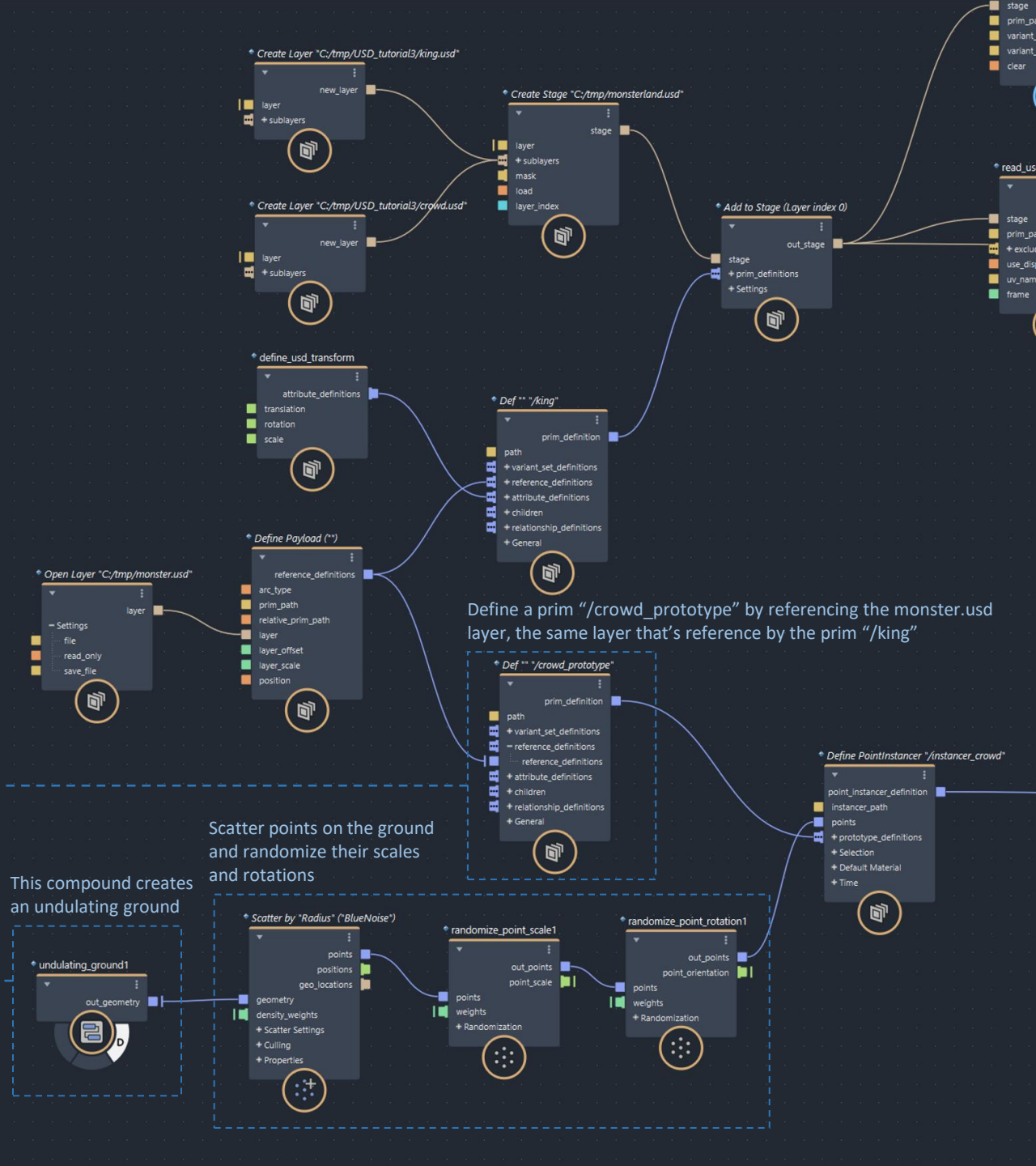
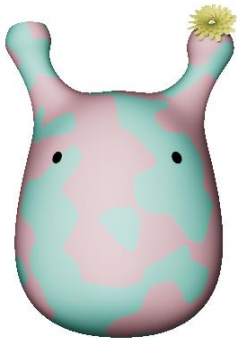
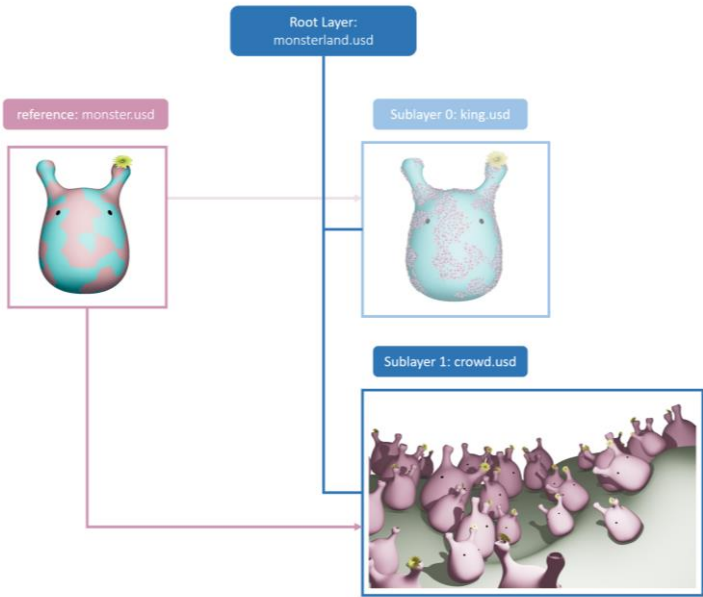
```
def "king" (
    prepend payload = @c:/tmp/monster.usd@</monster>
)
{
    float3 xformOp:rotateXYZ = (0, 0, 0)
    float3 xformOp:scale = (1.4, 1.4, 1.4)
    float3 xformOp:translate = (0, 0, 0.5)
    token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ", "xformOp:scale"]

    over "body"
    {
        over "noise"
        {
            color3f[] primvars:displayColor = [(0.0539, 0.2377, 0.3859)] ( ←
                interpolation = "constant"
            )
        }
    }
}

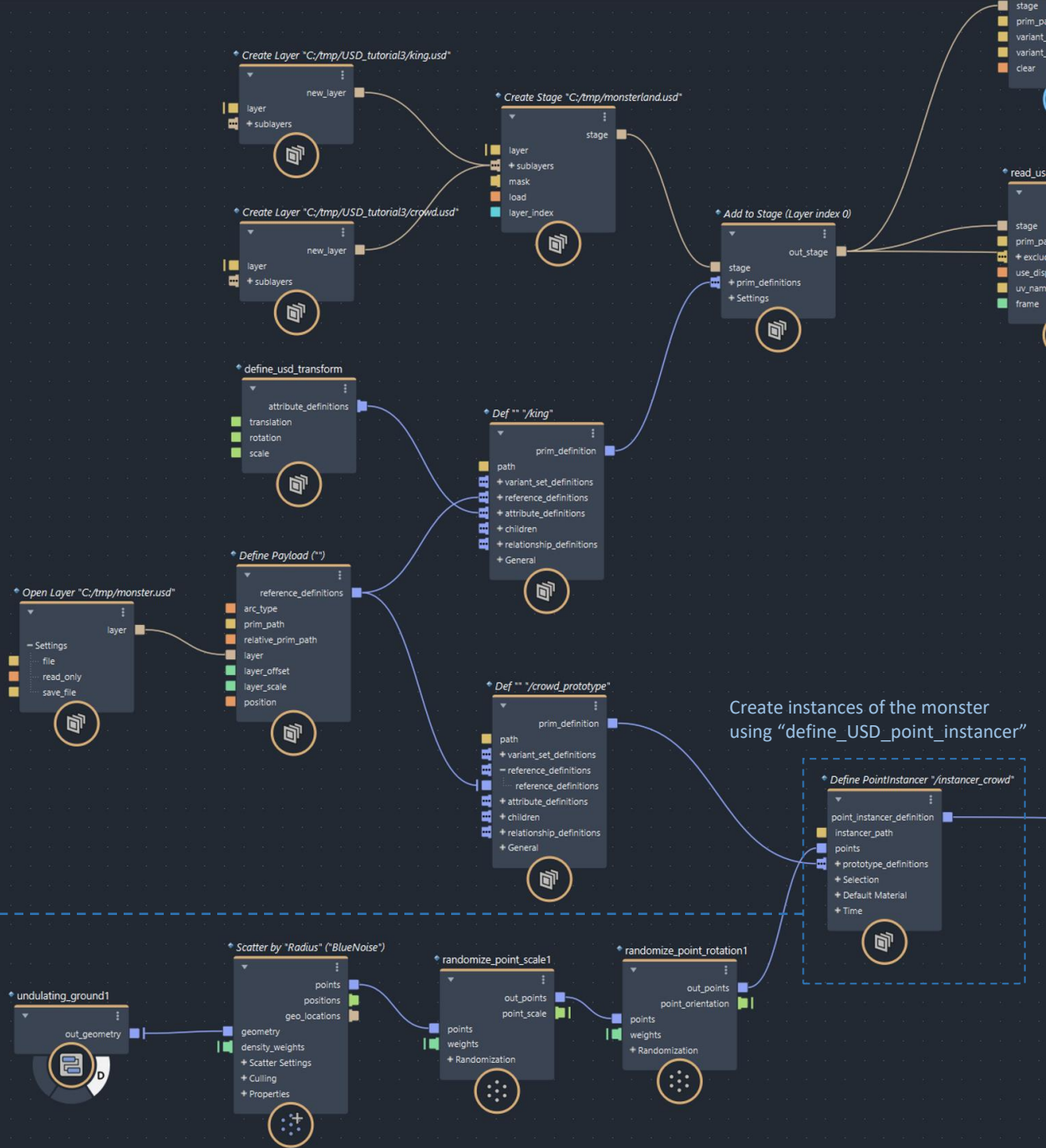
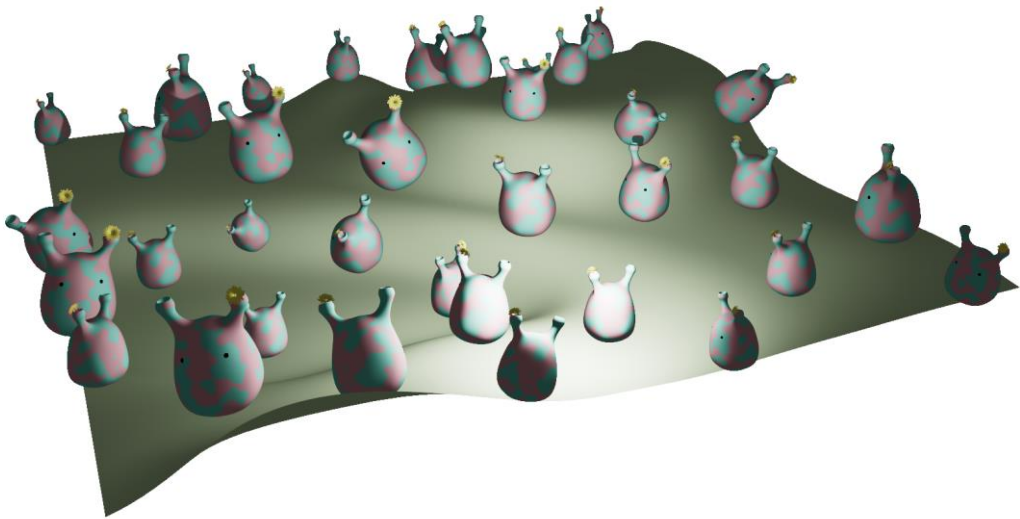
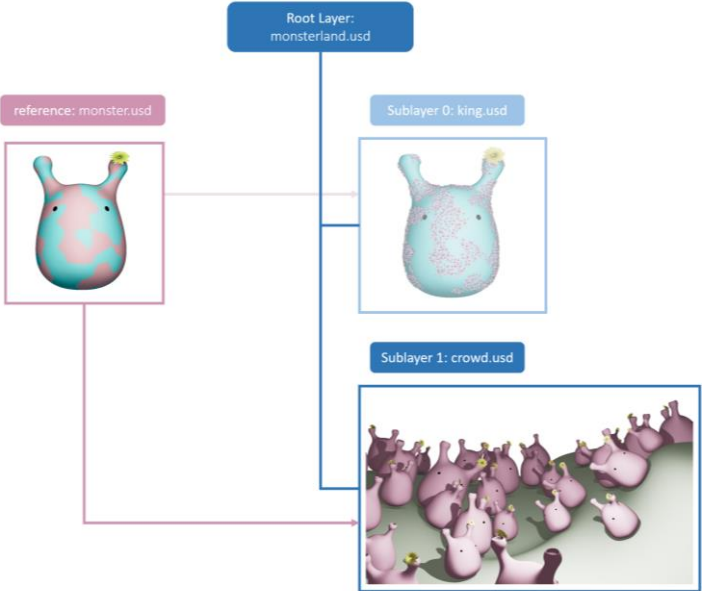
def PointInstancer "instancer_decoration"
{
    quath[] orientations = [(0.896484, -0.256104, -0.309814, 0.186401), (0.92334, -0.0771484,
point3f[] positions = [(-0.68970585, 3.8572726, 0.094121285), (-0.56112796, 3.9480093, 0.
int[] protoIndices = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
prepend rel prototypes = </instancer_decoration/prototypes/decoration>
float3[] scales = [(0.9653187, 0.9653187, 0.9653187), (1.0275096, 1.0275096, 1.0275096),

def "prototypes"
{
    def "decoration" (
        prepend payload = @c:/tmp/myosotis.usd@</myosotis>
    )
    {}
}
}
```

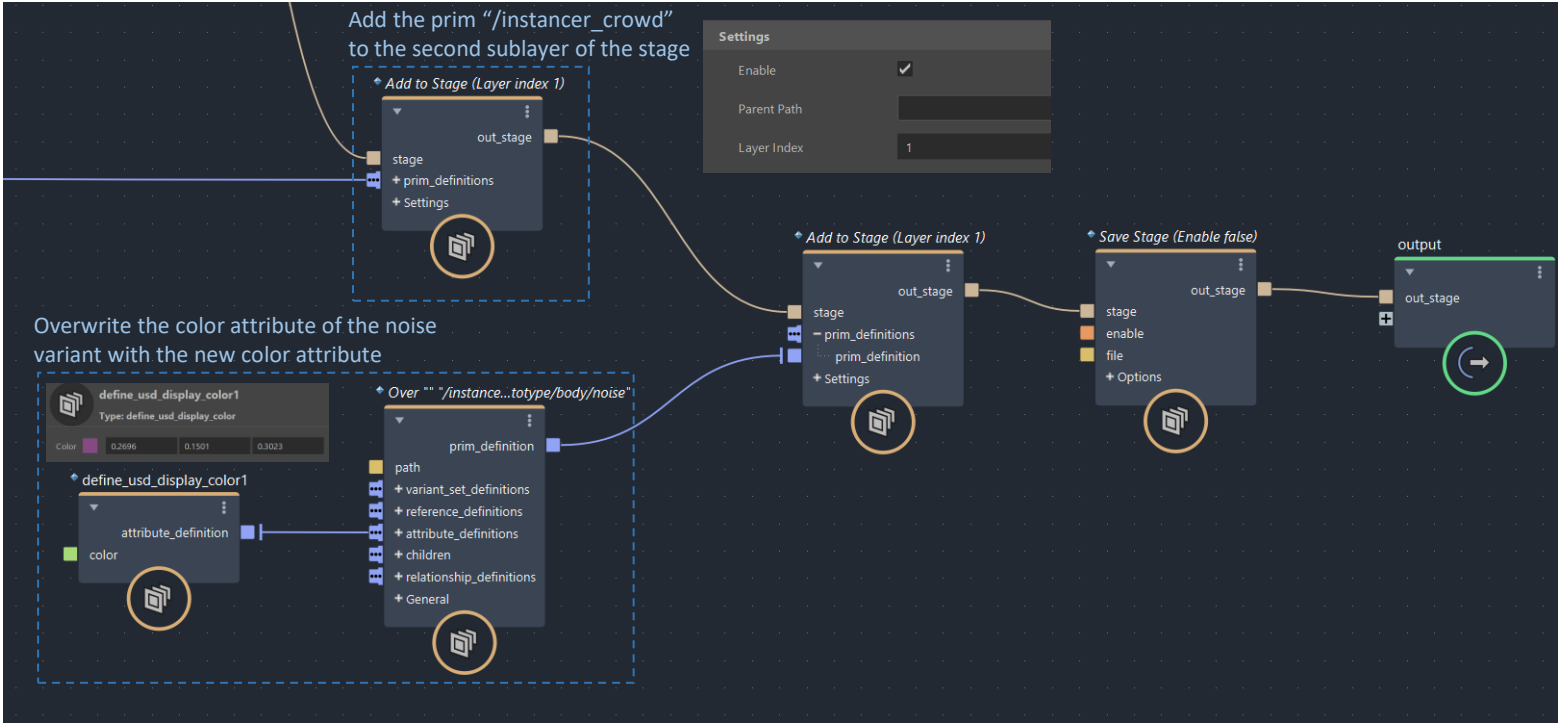
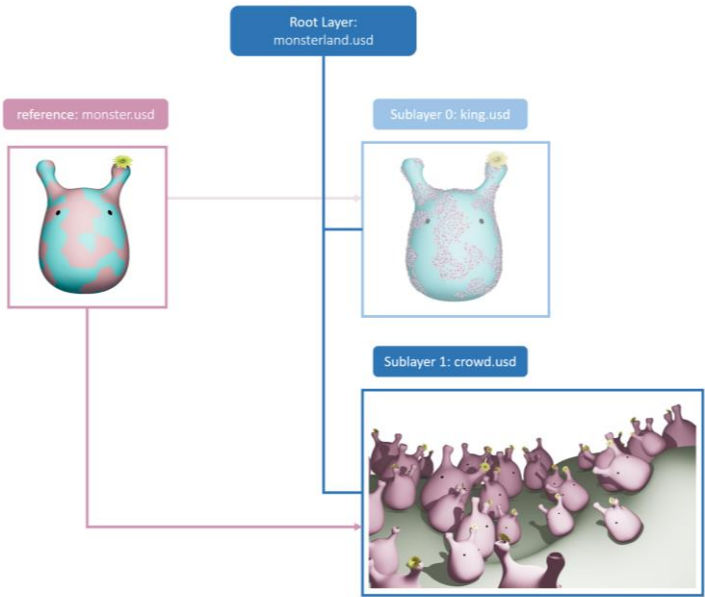

Create the crowd



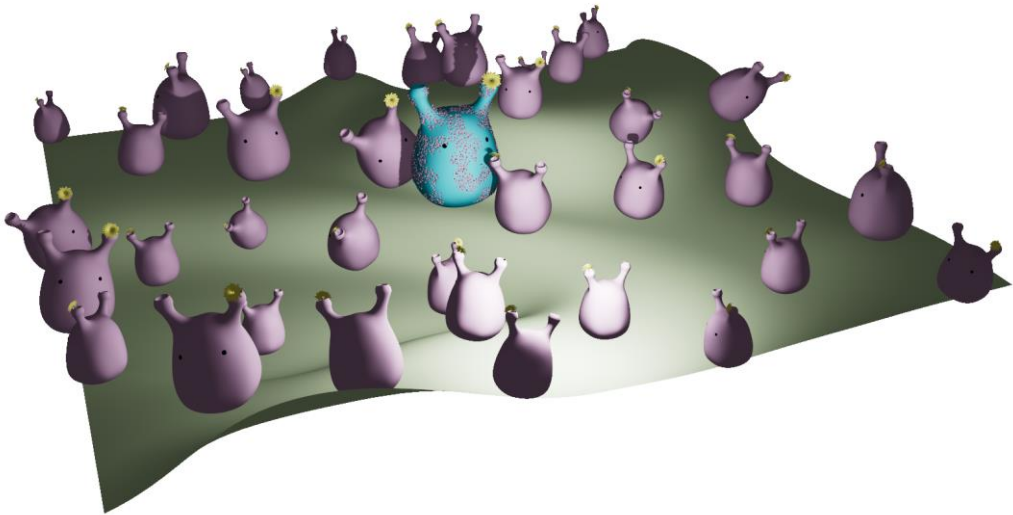
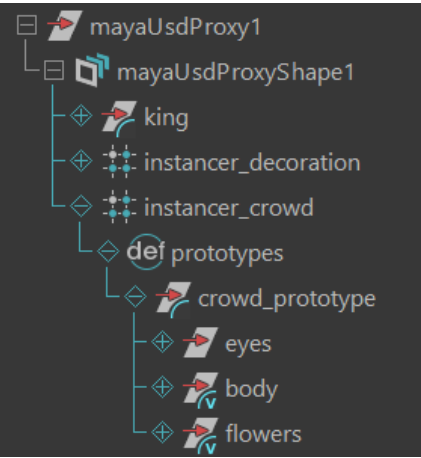
Create the crowd

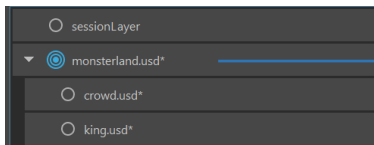


Create the crowd



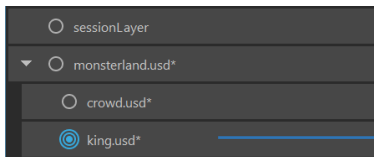
Outliner





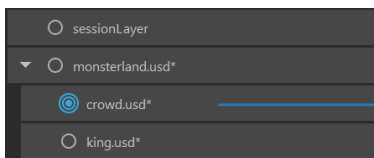
Root layer: monsterland.usd

```
// USD Layer identifier: anon:000002683DD8E530:monsterland.usd
// Real Path:
// #usda 1.0
// (
//   defaultPrim = "king"
//   subLayers = [
//     @anon:000002683DD8EC30:crowd.usd@,
//     @anon:000002683DD8D30:king.usd@
//   ]
// )
```



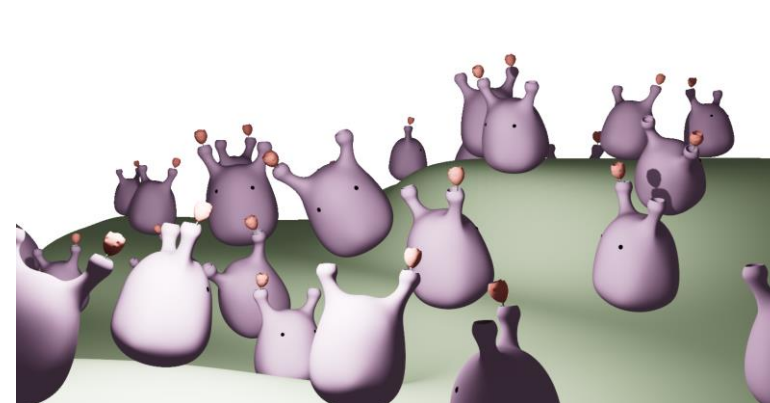
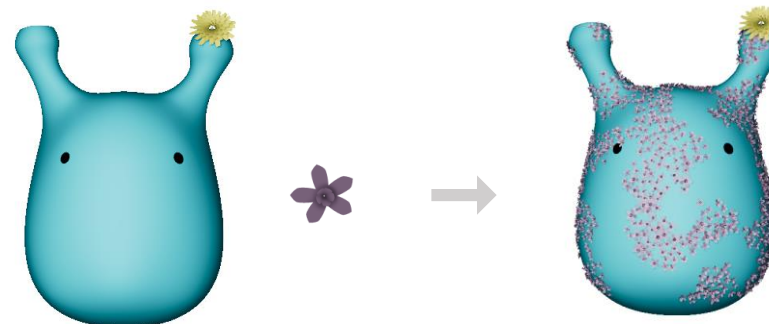
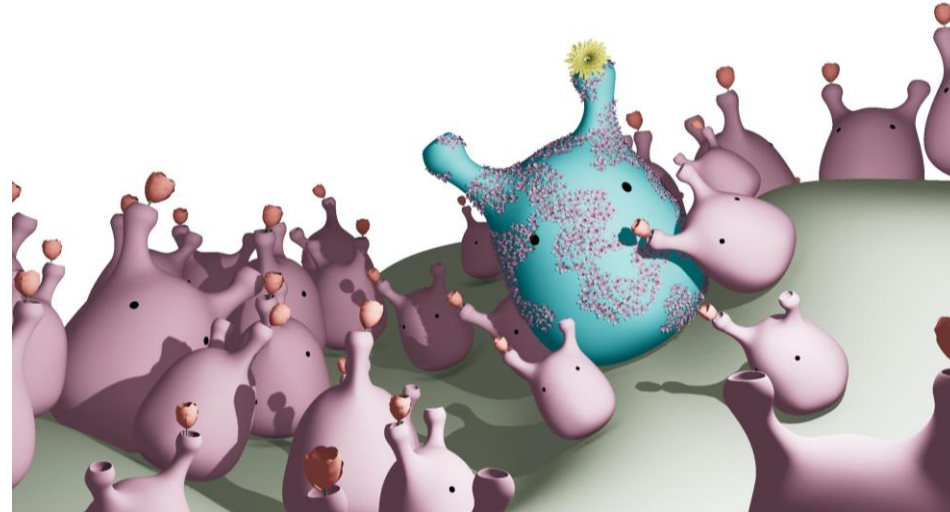
Sublayer 0: king.usd

```
// def "king" (
//   prepend payload = @c:/tmp/monster.usd@</monster>
// )
// {
//   float3 xformOp:rotateXYZ = (0, 0, 0)
//   float3 xformOp:scale = (1.4, 1.4, 1.4)
//   float3 xformOp:translate = (0, 0, 0.5)
//   token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ", "xformOp:scale"]
//
//   over "body"
//   {
//     over "noise"
//     {
//       color3f[] primvars:displayColor = [(0.0539, 0.2377, 0.3859)] (
//         interpolation = "constant"
//       )
//     }
//   }
// }
//
// def PointInstancer "instancer_decoration"
// {
//   quath[] orientations = [(0.896484, -0.256104, -0.309814, 0.186401), (0.92334, -0.077
//   point3f[] positions = [(-0.68970585, 3.8572726, 0.094121285), (-0.56112796, 3.948009
//   int[] protoIndices = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
//   prepend rel prototypes = </instancer_decoration/prototypes/decoration>
//   float3[] scales = [(0.9653187, 0.9653187, 0.9653187), (1.0275096, 1.0275096, 1.02750
//
//   def "prototypes"
//   {
//     def "decoration" (
//       prepend payload = @c:/tmp/myosotis.usd@</myosotis>
//     )
//     {
//       {
//         }
//       }
//     }
//   }
// }
```



Sublayer 1: crowd.usd

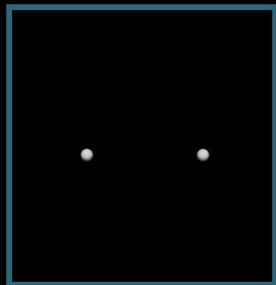
```
// def PointInstancer "instancer_crowd"
// {
//   quath[] orientations = [(0.959473, 0.223999, 0.0997925, 0.139282), (0.996582, -0.044
//   point3f[] positions = [(-14.755994, -1.7497505, -4.516179), (-14.851114, -2.3495574,
//   int[] protoIndices = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
//   prepend rel prototypes = </instancer_crowd/prototypes/crowd_prototype>
//   float3[] scales = [(0.9653187, 0.9653187, 0.9653187), (1.0275096, 1.0275096, 1.02750
//
//   def "prototypes"
//   {
//     def "crowd_prototype" (
//       prepend payload = @c:/tmp/monster.usd@</monster>
//     )
//     {
//       over "body"
//       {
//         over "noise"
//         {
//           color3f[] primvars:displayColor = [(0.2696, 0.1501, 0.3023)] (
//             interpolation = "constant"
//           )
//         }
//       }
//     }
//   }
// }
```



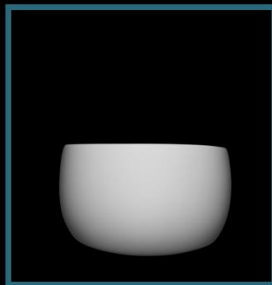
USD Material Binding



/monster/geometries/
body

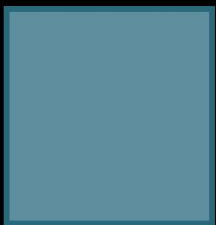


/monster/geometries/
eyes

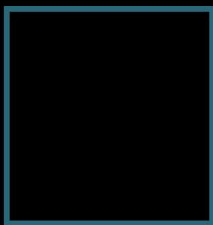


/monster/geometries/
sweater

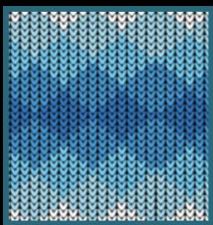
/monster/geometries



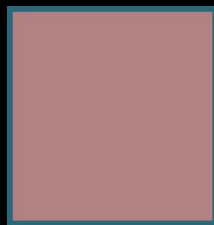
/monster/materials/
blue



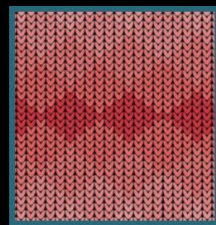
/monster/materials/
black



/monster/materials/
textile_blue

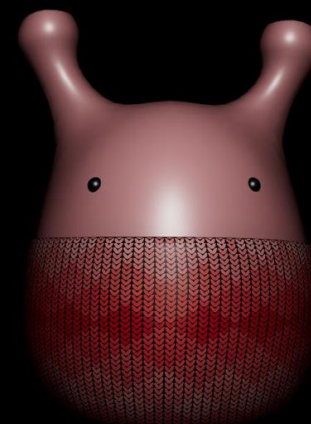
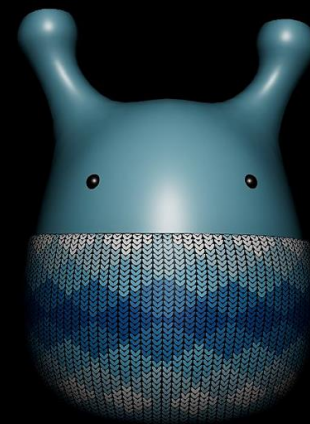


/monster/materials/
red

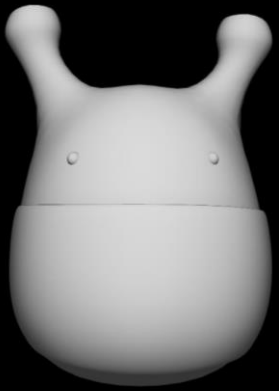


/monster/materials/
textile_red

/monster/materials



/monster



monster_no_material.usd

Path: /textile_blue

Base

Diffuse Color: 0 0 0

Texture File Path: -and-white-pattern.jpg

Emissive Color: 0 0 0

Specular

Clearcoat: 0

Index Of Refraction: 1.52

Metallic: 0

Opacity: 1

Roughness: 0.46

Path to the texture image

Path: /blue

Base

Diffuse Color: 0.268 0.363 0.457

Texture File Path:

Emissive Color: 0 0 0

Specular

Clearcoat: 0

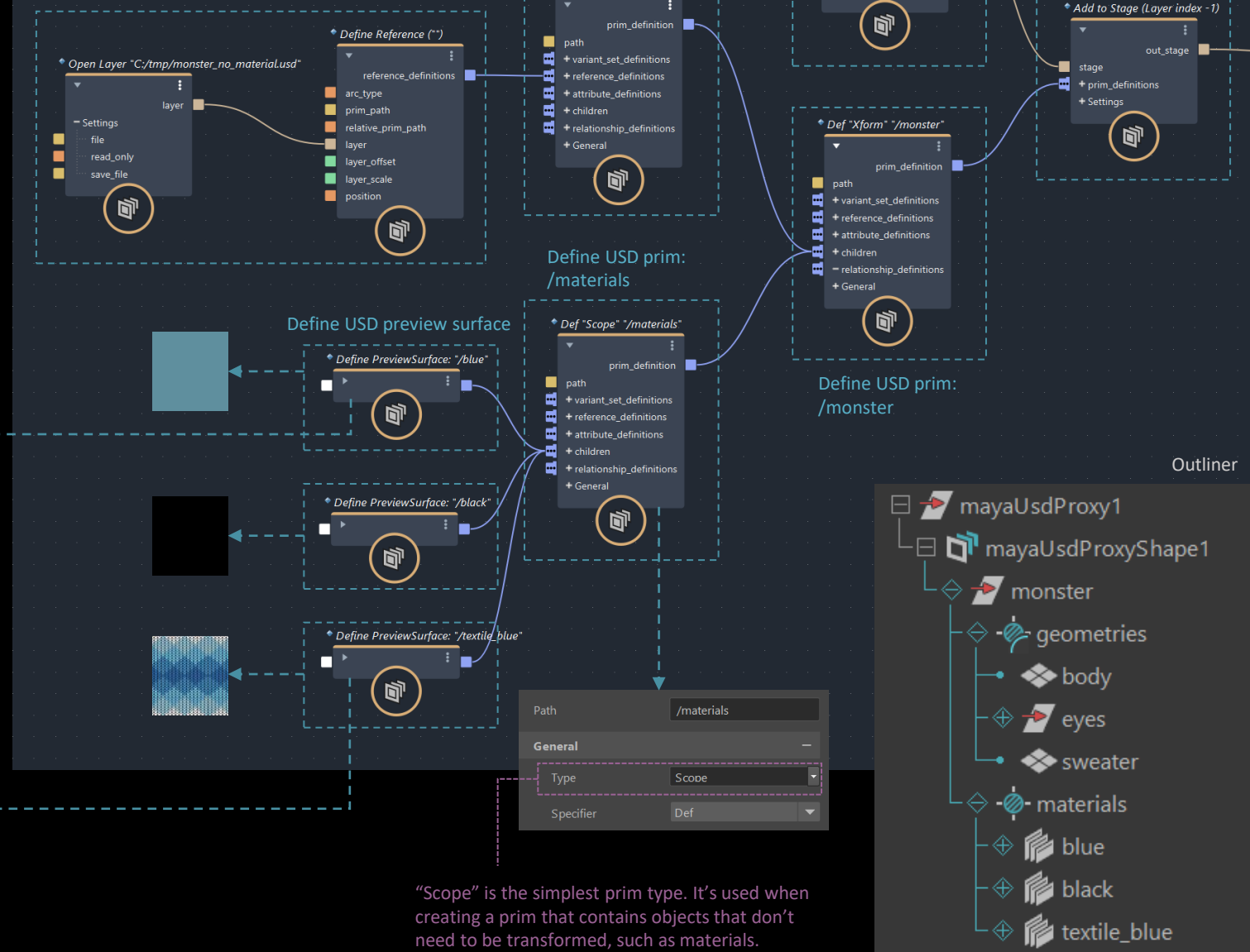
Index Of Refraction: 1.52

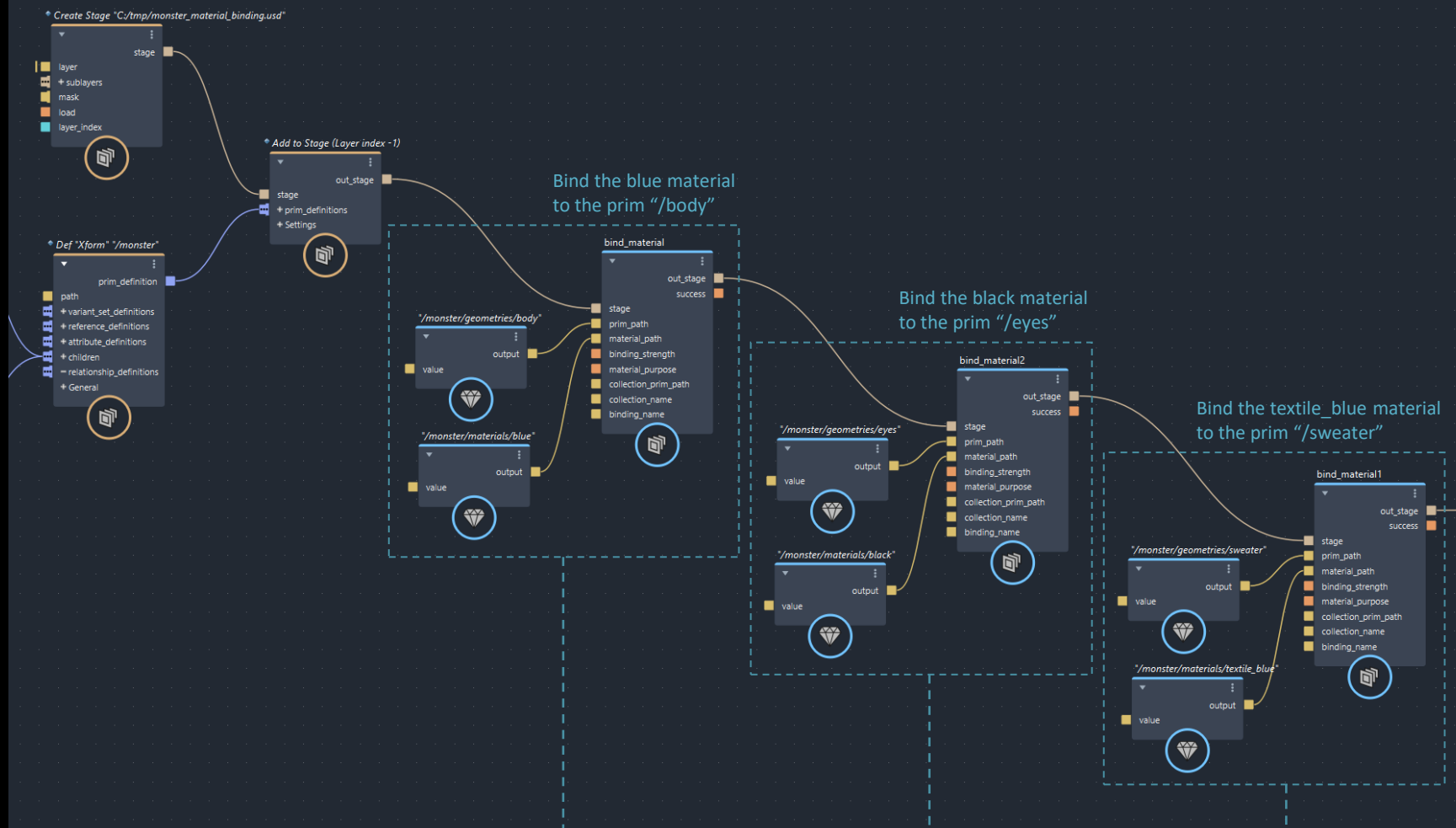
Metallic: 0

Opacity: 1

Roughness: 0.46

Reference the "monster_no_material.usd" layer
This layer contains the body, eyes and sweater of the monster without any surface material.





Bind material

bind_material
 Type: bind_material

Stage:

Prim Path:

Material Path:

Binding Strength:

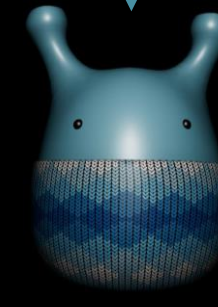
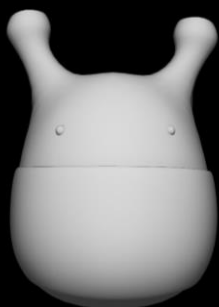
Material Purpose:

Collection Prim Path:

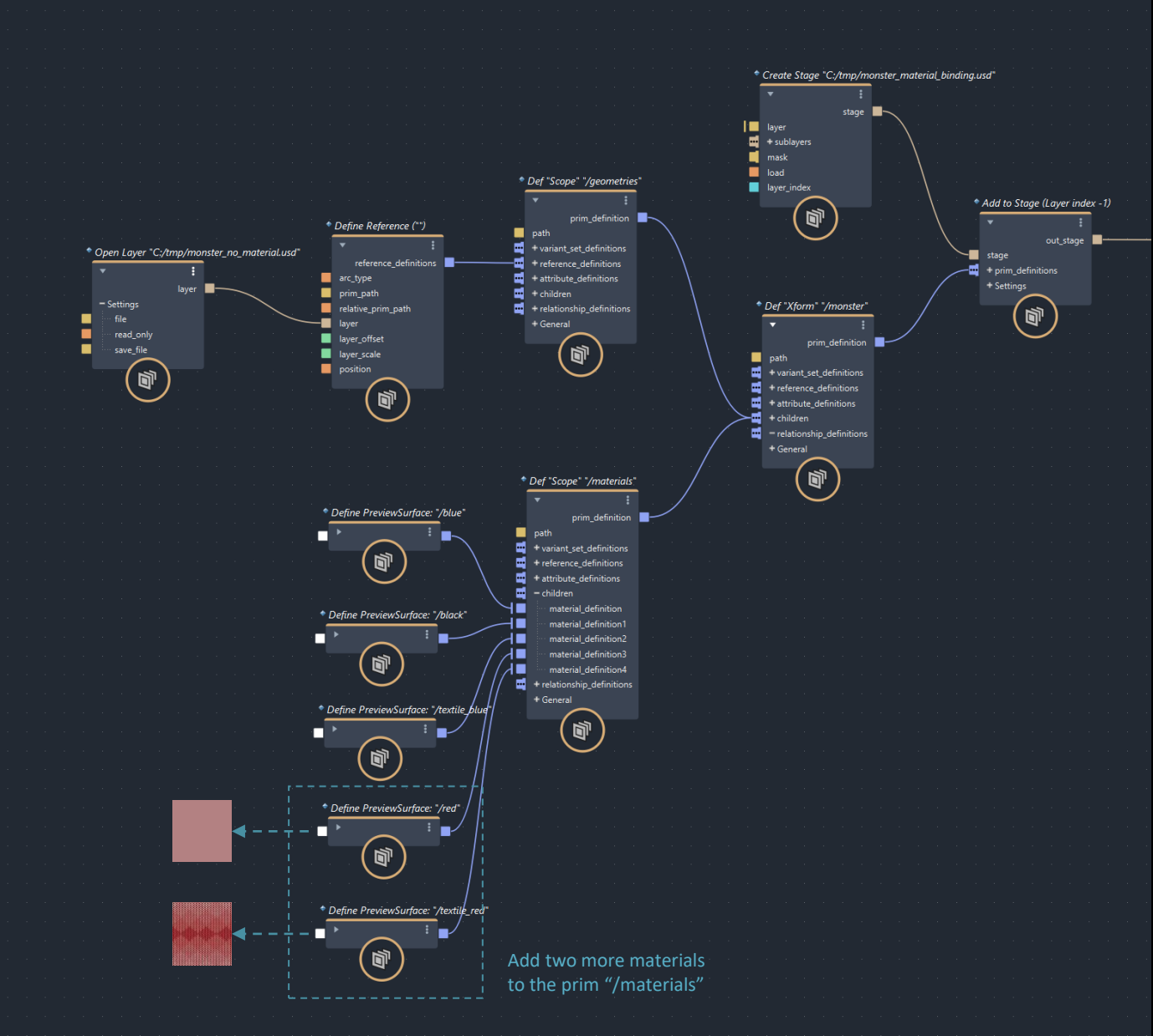
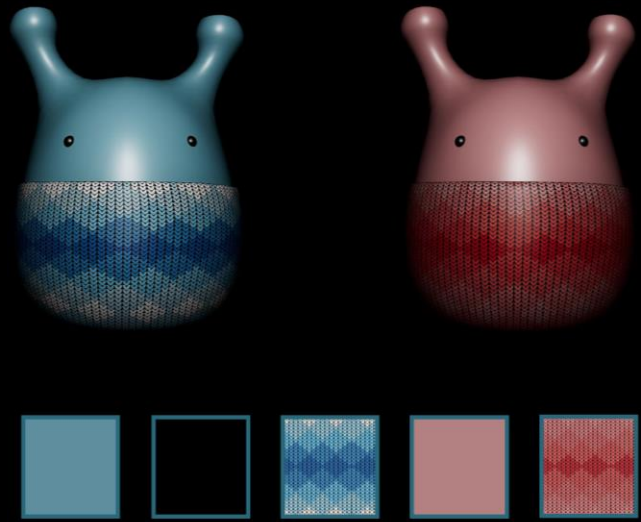
Collection Name:

Binding strength:

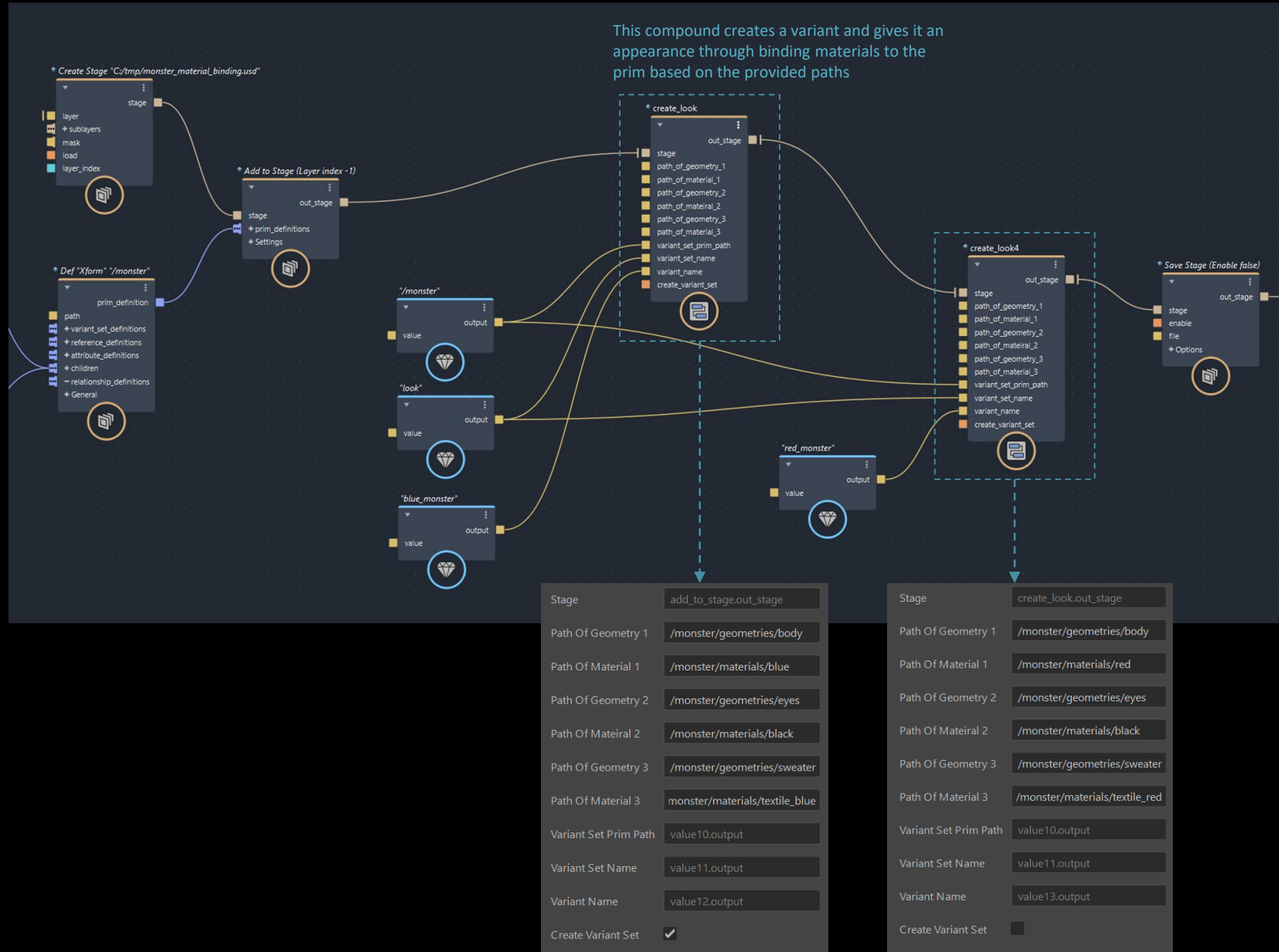
- Stronger than descendants: a child prim always inherit the material of the parent prim
- Weaker than descendants: a child prim doesn't inherit the material of the parent prim
- Fall back strength: the material of a parent prim doesn't overwrite the material assigned to its child prim



Create a variant set of two looks



Create a variant set of two looks



Create a variant set of two looks

