Custom Gaussian Filter

This is example code of a custom gaussian filter.

```
#include <ai.h>
#include <string.h>

AI_FILTER_NODE_EXPORT_METHODS(CustomGaussianFilterMtd);

static AtString WIDTH("width");

node_parameters
{
   AiParameterFlt("width", 2.0f);
}

node_initialize
{
   AiFilterInitialize(node, false, NULL, NULL);
}

node_update
{
   AiFilterUpdate(node, AiNodeGetFlt(node, WIDTH));
}

node_finish
{
   AiFilterDestroy(node);
}

filter_output_type
{
   switch (input_type)
   {
      case AI_TYPE_RGBA:
         return AI_TYPE_RGBA;
      default:
         return AI_TYPE_NONE;
   }
}

filter_pixel
{
   const float width = AiNodeGetFlt(node, WIDTH);

   float aweight = 0.0f;
   AtRGBA avalue = AI_RGBA_ZERO;

   while (AiAOVSampleIteratorGetNext(iterator))
   {
      // take into account adaptive sampling
      float inv_density = AiAOVSampleIteratorGetInvDensity(iterator);
      if (inv_density <= 0.f)
         continue;

      // determine distance to filter center
      const AtPoint2& offset = AiAOVSampleIteratorGetOffset(iterator);
      const float r = SQR(2 / width) * (SQR(offset.x) + SQR(offset.y));
      if (r > 1.0f)
         continue;
      // gaussian filter weight
      const float weight = fast_exp(2 * -r) * inv_density;

      // accumulate weights and colors
      avalue += weight * AiAOVSampleIteratorGetRGBA(iterator);
```

```
      aweight += weight;
   }

   // compute final filtered color
   if (aweight != 0.0f)
      avalue /= aweight;

   *((AtRGBA*)data_out) = avalue;
}

node_loader
{
   if (i>0)
      return false;

   node->methods      = CustomGaussianFilterMtd;
   node->output_type  = AI_TYPE_NONE;
   node->name         = "custom_gaussian_filter";
   node->node_type    = AI_NODE_FILTER;
   strcpy(node->version, AI_VERSION);

   return true;
}
```