The flow of `radii`

- meaning of `radii`: semi-major axis and semi-minor axis of ellipse
  - longest radius of the ellipse
  - shortest radius of the ellipse
- defined in `gaussian_renderer/__init__.py`:

```
rendered_image, radii = rasterizer(
        means3D = means3D,
        means2D = means2D,
        shs = shs,
        colors_precomp = colors_precomp,
        opacities = opacity,
        scales = scales,
        rotations = rotations,
        cov3D_precomp = cov3D_precomp)
```

- obtain from rasterizer:

```
rasterizer = GaussianRasterizer(raster_settings=raster_settings)
```

- invoke `submodules\diff-gaussian-rasterization\diff_gaussian_rasterization\__init__.py`

```
class GaussianRasterizer(nn.Module):
```

- invoke C++/CUDA rasterization routine

```
def rasterize_gaussians(
    means3D,
    means2D,
    sh,
    colors_precomp,
    opacities,
    scales,
    rotations,
    cov3Ds_precomp,
    raster_settings,
):
    return _RasterizeGaussians.apply(
        means3D,
        means2D,
        sh,
        colors_precomp,
        opacities,
        scales,
        rotations,
        cov3Ds_precomp,
        raster_settings,
    )
...
class _RasterizeGaussians(torch.autograd.Function):
    ...
    # Invoke C++/CUDA rasterizer
    if raster_settings.debug:
        cpu_args = cpu_deep_copy_tuple(args) # Copy them before they can be corrupted
        try:
            num_rendered, color, radii, geomBuffer, binningBuffer, imgBuffer = _C.rasterize_gaussians(*args)
        except Exception as ex:
            torch.save(cpu_args, "snapshot_fw.dump")
            print("\nAn error occured in forward. Please forward snapshot_fw.dump for debugging.")
            raise ex
    else:
        num_rendered, color, radii, geomBuffer, binningBuffer, imgBuffer = _C.rasterize_gaussians(*args)
```

- connect python with C++/CUDA in `submodules\diff-gaussian-rasterization\ext.cpp`

```
PYBIND11_MODULE(TORCH_EXTENSION_NAME, m) {
  m.def("rasterize_gaussians", &RasterizeGaussiansCUDA);
  m.def("rasterize_gaussians_backward", &RasterizeGaussiansBackwardCUDA);
  m.def("mark_visible", &markVisible);
}
```

- go to `RasterizeGaussiansCUDA` in `submodules\diff-gaussian-rasterization\rasterize_points.cu`, go to `submodules\diff-gaussian-rasterization\cuda_rasterizer`, radii is also passed to correspounding functions

```
std::tuple<int, torch::Tensor, torch::Tensor, torch::Tensor, torch::Tensor, torch::Tensor>
RasterizeGaussiansCUDA(
    const torch::Tensor& background,
    const torch::Tensor& means3D,
    const torch::Tensor& colors,
    const torch::Tensor& opacity,
    const torch::Tensor& scales,
    const torch::Tensor& rotations,
    const float scale_modifier,
    const torch::Tensor& cov3D_precomp,
    const torch::Tensor& viewmatrix,
    const torch::Tensor& projmatrix,
    const float tan_fovx,
    const float tan_fovy,
    const int image_height,
    const int image_width,
    const torch::Tensor& sh,
    const int degree,
    const torch::Tensor& campos,
    const bool prefiltered,
    const bool debug)
{
    ...
    rendered = CudaRasterizer::Rasterizer::forward(
        geomFunc,
        binningFunc,
        imgFunc,
        P, degree, M,
        background.contiguous().data<float>(),
        W, H,
        means3D.contiguous().data<float>(),
        sh.contiguous().data_ptr<float>(),
        colors.contiguous().data<float>(),
        opacity.contiguous().data<float>(),
        scales.contiguous().data_ptr<float>(),
        scale_modifier,
        rotations.contiguous().data_ptr<float>(),
        cov3D_precomp.contiguous().data<float>(),
        viewmatrix.contiguous().data<float>(),
        projmatrix.contiguous().data<float>(),
        campos.contiguous().data<float>(),
        tan_fovx,
        tan_fovy,
        prefiltered,
        out_color.contiguous().data<float>(),
        radii.contiguous().data<int>(), // radii passed to forward.cu
        debug);
}
```

- `CudaRasterizer::Rasterizer::forward` defined in `submodules\diff-gaussian-rasterization\cuda_rasterizer\rasterizer_impl.cu` and will call functions (i.e., preprocessCUDA, renderCUDA) in `forward.cu`
- will initialize radii[idx] = 0, if not in frustum, will directly return and radii will remain 0.
- Gaussians with radii = 0 will not pass the visibility_filter.

```
// submodules\diff-gaussian-rasterization\cuda_rasterizer\forward.cu
template<int C>
__global__ void preprocessCUDA(int P, int D, int M,
    const float* orig_points,
    const glm::vec3* scales,
    const float scale_modifier,
    const glm::vec4* rotations,
    const float* opacities,
    const float* shs,
    bool* clamped,
    const float* cov3D_precomp,
    const float* colors_precomp,
    const float* viewmatrix,
    const float* projmatrix,
    const glm::vec3* cam_pos,
    const int W, int H,
    const float tan_fovx, float tan_fovy,
    const float focal_x, float focal_y,
    int* radii,
    float2* points_xy_image,
    float* depths,
    float* cov3Ds,
    float* rgb,
    float4* conic_opacity,
    const dim3 grid,
    uint32_t* tiles_touched,
    bool prefiltered)
{
    auto idx = cg::this_grid().thread_rank();
    if (idx >= P)
        return;

    // Initialize radius and touched tiles to 0. If this isn't changed,
    // this Gaussian will not be processed further.
    radii[idx] = 0;
    tiles_touched[idx] = 0;

    // Perform near culling, quit if outside.
    float3 p_view;
    if (!in_frustum(idx, orig_points, viewmatrix, projmatrix, prefiltered,
p_view))
        return;
...
}
```