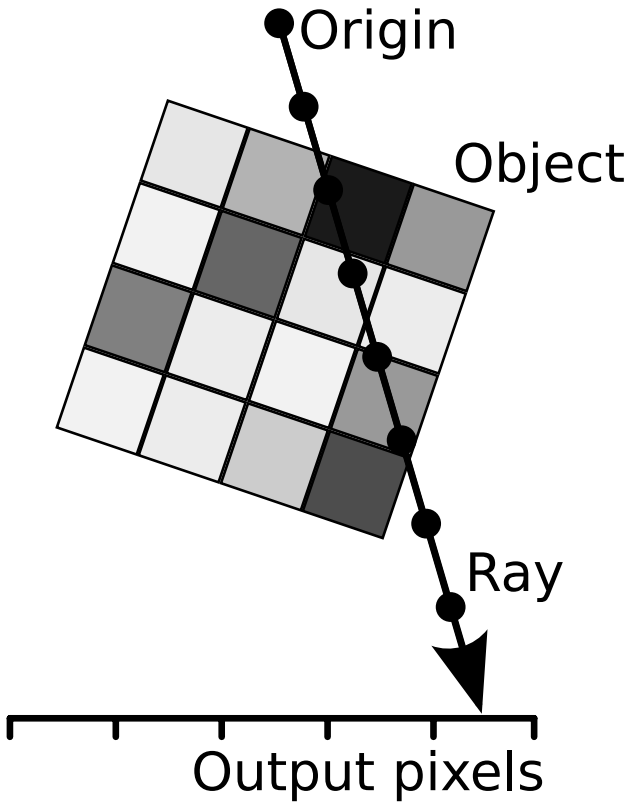
OpenCL exercise 5: Volume rendering

Kaicong Sun

Volume rendering



- Ray goes from origin to the output pixels
- Values of object (= input data) along the ray are summed up
- If value is not taken in the middle of a pixel, trilinear interpolation is used (bilinear in 2D-case)
- Sum of the values is value for output pixel
- Values outside the input object = 0

Task

- GPU implementation of 3D volume rendering
  - Use 3D image object for input data.
  - Each pixel on the display is a work-item.
- Profiling code which prints the CPU time / GPU time / memory transfer and speedups.
  - For memory transfer: Only time for transfering output data

Hints on Host

```
//The inverse view matrix (put camera in world space)
d_invViewMatrix is a floating points matrix with
16 elements (defined in render() function).

//For writing 3D image:
queue.enqueueWriteImage(d_input, true, origin, region,
countX * sizeof (float), countX * countY * sizeof (float),
(void*) h_input, NULL, &copyToDev);

//For launching the kernel:
work-group size can also be cl::NullRange in which case
the OpenCL implementation will determine how to break the
global work-items into appropriate work-group instances.
```

Hints on Device

```
// Data type and operations:
OpenCL library has defined float3, min(), max(),
normalize(), dot()

//For getting image size:
float3 boxMax = (float3)get_image_width(d_input),
get_image_height(d_input), get_image_depth(d_input)

// using CLK_FILTER_LINEAR in sampler to have
interpolated value (note for 3D image, coordinates are
float4/int4):
float sample = read_imagef(d_input, sampler,
(float4)(pos, 0)).x
```