Macoun'10

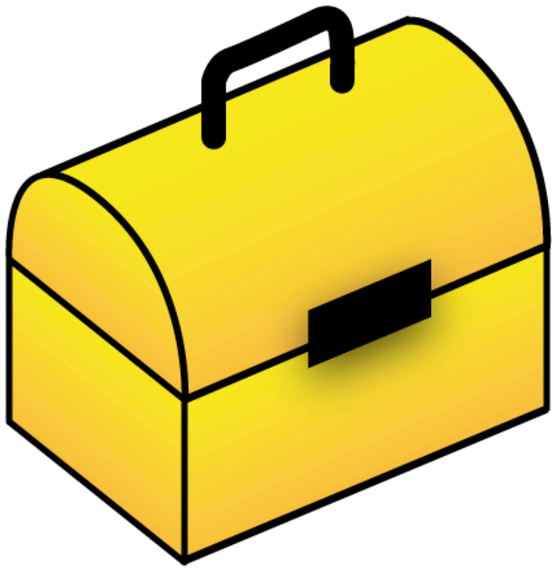
Umdenken in OpenCL

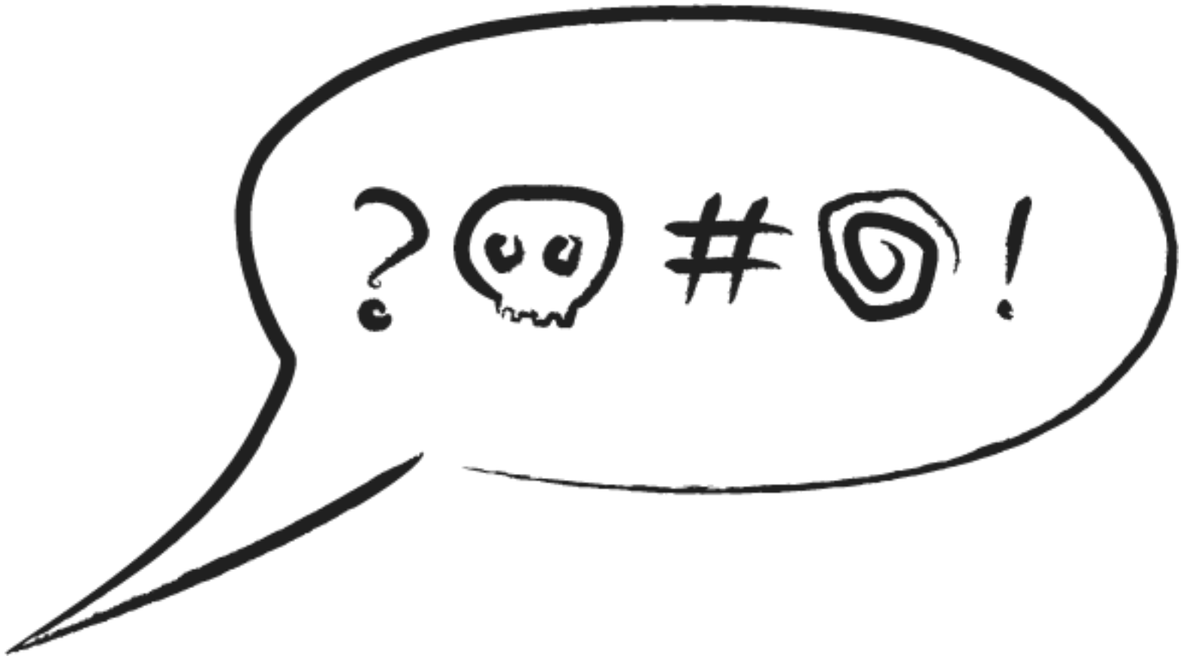
Eberhard Rensch

Ablauf

- Vorspeise: Grundlagen von OpenCL
- Hauptgericht

Was ist OpenCL

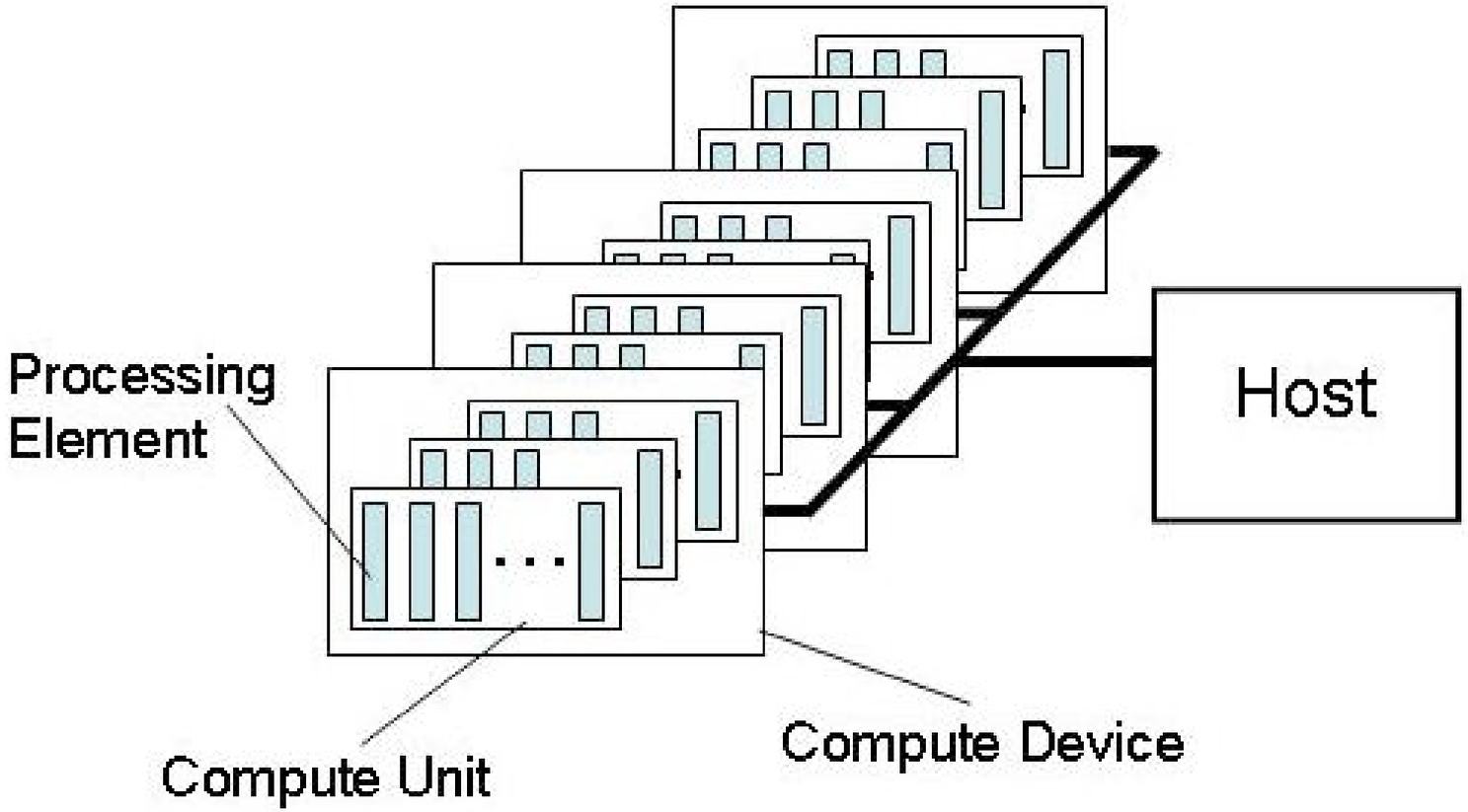




Was ist OpenCL

- http://www.khronos.org/opencl/
- OpenGL, OpenAL, OpenCL
- OpenCL-Programm Infrastruktur erinnert OpenGL GLSL-Shader-Programm Infrastruktur

OpenCL Plattform



Runtime OpenCL

```
cl_device_id device_id;
err = clGetDeviceIDs(NULL,
             useGPU ? CL_DEVICE_TYPE_GPU : CL_DEVICE_TYPE_CPU,
             1, &device_id, NULL);
```

Runtime OpenCL

```
cl_device_id device_id;
err = clGetDeviceIDs(NULL,
                     useGPU ? CL_DEVICE_TYPE_GPU : CL_DEVICE_TYPE_CPU,
                     1, &device_id, NULL);
```

Runtime OpenCL

```
cl_device_id device_id;
err = clGetDeviceIDs(NULL,
                     useGPU ? CL_DEVICE_TYPE_GPU : CL_DEVICE_TYPE_CPU,
                     1, &device_id, NULL);
```

Demo

Runtime OpenCL

```
cl_device_id device_id;
err = clGetDeviceIDs(NULL,
            useGPU ? CL_DEVICE_TYPE_GPU : CL_DEVICE_TYPE_CPU,
            1, &device_id, NULL);
```

```
cl_context context = clCreateContext(0, 1, &device_id, NULL, NULL, &err);
```

Runtime OpenCL

```
cl_device_id device_id;
err = clGetDeviceIDs(NULL,
            useGPU ? CL_DEVICE_TYPE_GPU : CL_DEVICE_TYPE_CPU,
            1, &device_id, NULL);
```

```
cl_context context = clCreateContext(0, 1, &device_id, NULL, NULL, &err);
```

Runtime OpenCL

```
cl_device_id device_id;
err = clGetDeviceIDs(NULL,
            useGPU ? CL_DEVICE_TYPE_GPU : CL_DEVICE_TYPE_CPU,
            1, &device_id, NULL);
```

```
cl_context context = clCreateContext(0, 1, &device_id, NULL, NULL, &err);
```

```
cl_command_queue queue = clCreateCommandQueue(context, device_id, 0, &err);
```

Runtime OpenCL

```
cl_program program = clCreateProgramWithSource(context, 1,
                                               (const char **) source, NULL, &err);
```

Runtime OpenCL

```
cl_program program = clCreateProgramWithSource(context, 1,
                                                (const char **) source, NULL, &err);
```

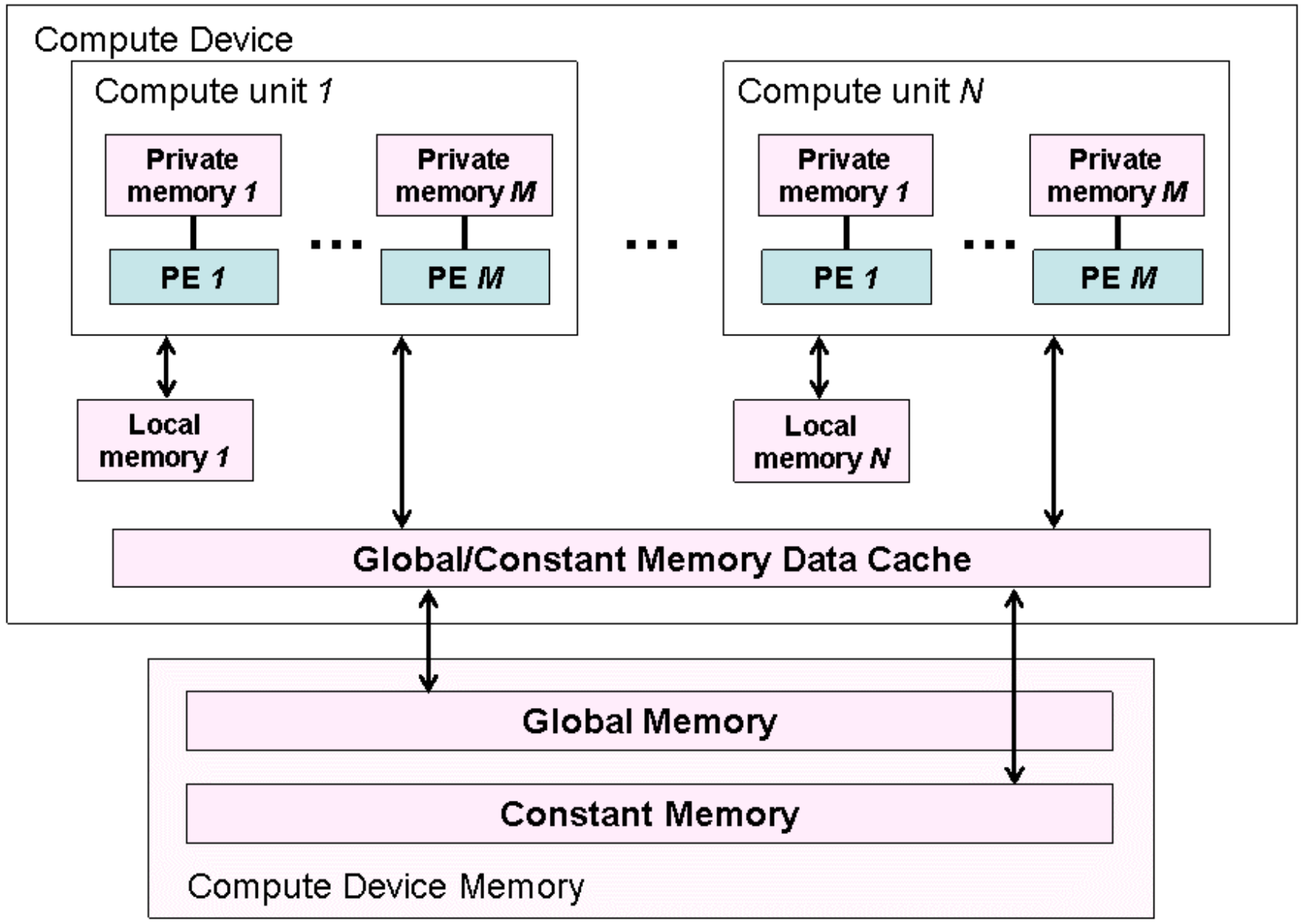
Runtime OpenCL

```
cl_program program = clCreateProgramWithSource(context, 1,
                                               (const char **) source, NULL, &err);
```

```
 err = clBuildProgram(program, 0, NULL, NULL, NULL, NULL);
```

Demo

Speichermodell OpenCL



Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

```
err = clEnqueueWriteBuffer(queue, buffer, CL_FALSE, 0,
                           sizeof(cl_float) * count, daten, 0, NULL, NULL);
```

Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

```
err = clEnqueueWriteBuffer(queue, buffer, CL_FALSE, 0,
                           sizeof(cl_float) * count, daten, 0, NULL, NULL);
```

Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

```
err = clEnqueueWriteBuffer(queue, buffer, CL_FALSE, 0,
                           sizeof(cl_float) * count, daten, 0, NULL, NULL);
```

Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

```
err = clEnqueueWriteBuffer(queue, buffer, CL_FALSE, 0,
                           sizeof(cl_float) * count, daten, 0, NULL, NULL);
```

```
err = clEnqueueReadBuffer( queue, outputBuffer, CL_TRUE, 0,
                         sizeof(cl_float) * count, ergebnisse, 0, NULL, NULL );
```

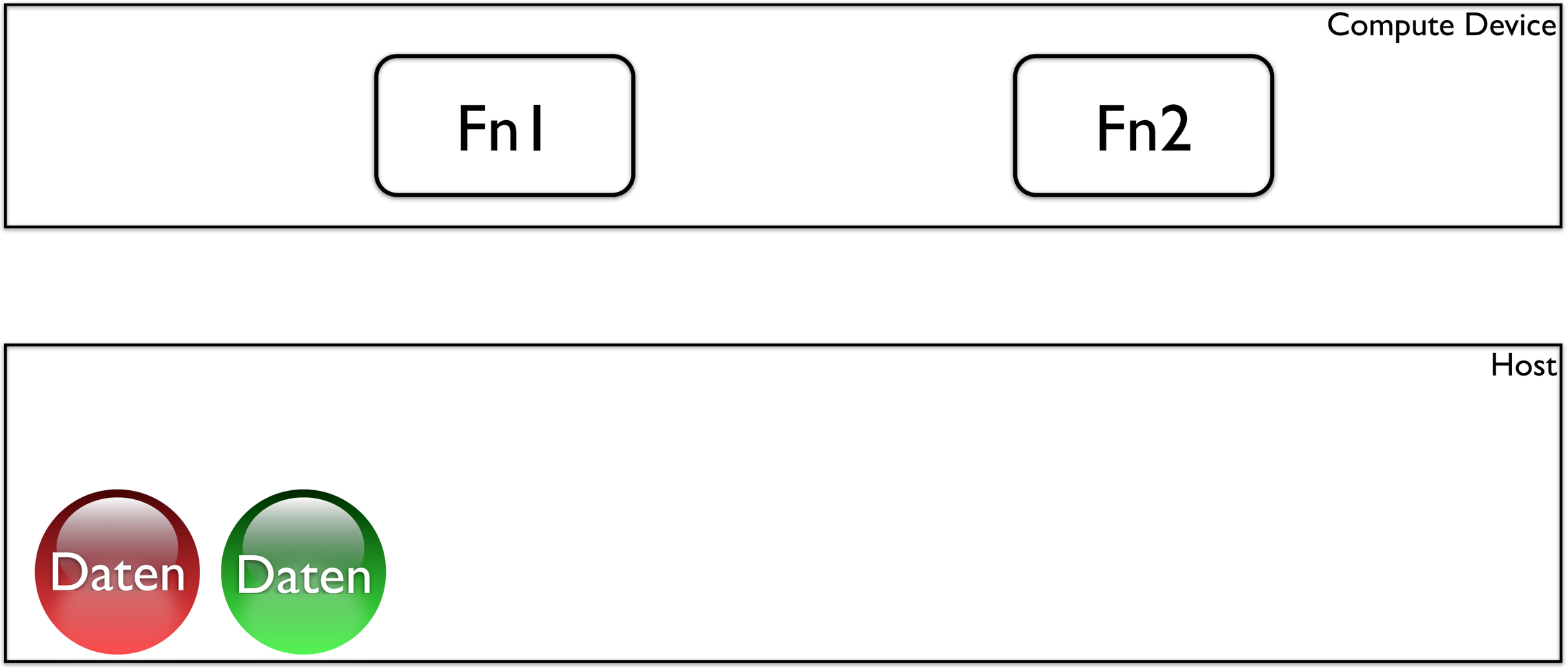
Speichermodell OpenCL

```
cl_mem buffer = clCreateBuffer(queue,  CL_MEM_READ_WRITE,
                                       sizeof(cl_float) * count, NULL, NULL);
```

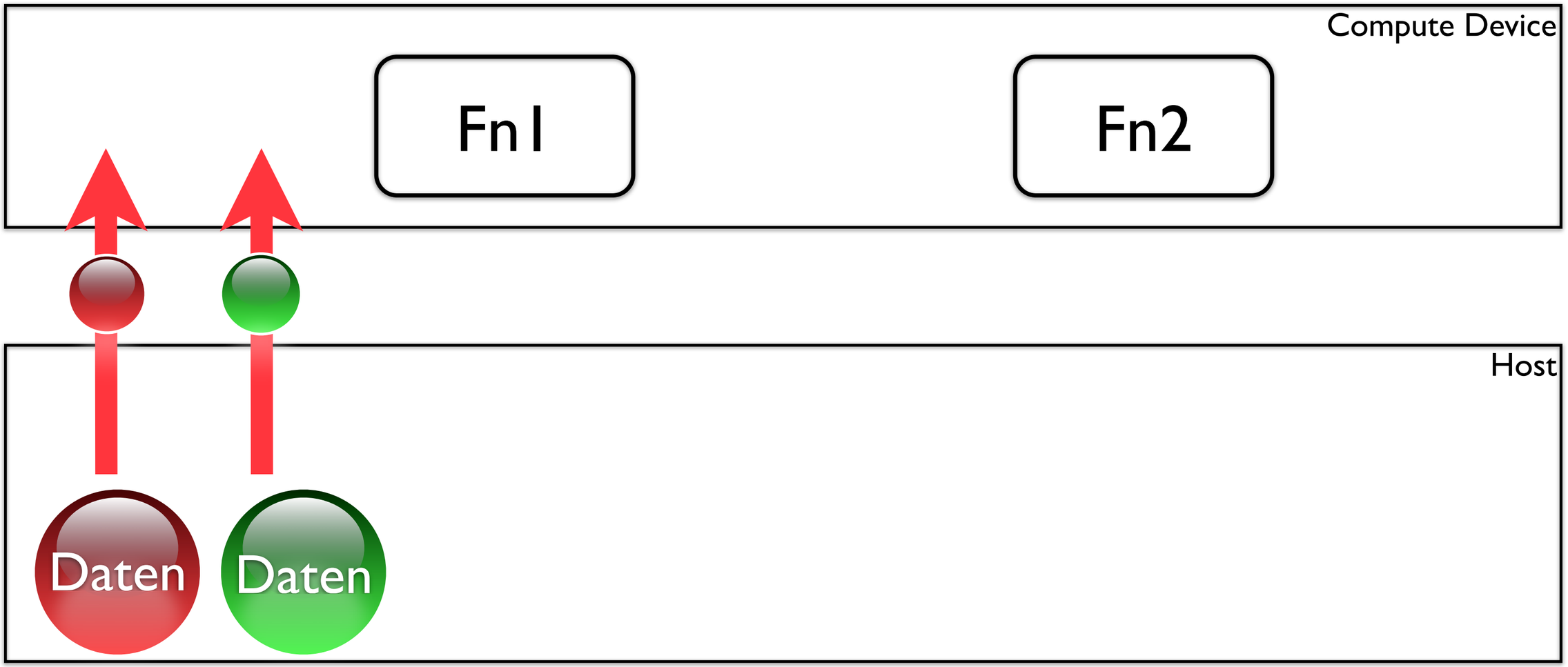
```
err = clEnqueueWriteBuffer(queue, buffer, CL_FALSE, 0,
                           sizeof(cl_float) * count, daten, 0, NULL, NULL);
```

```
err = clEnqueueReadBuffer( queue, outputBuffer, CL_TRUE, 0,
                         sizeof(cl_float) * count, ergebnisse, 0, NULL, NULL );
```

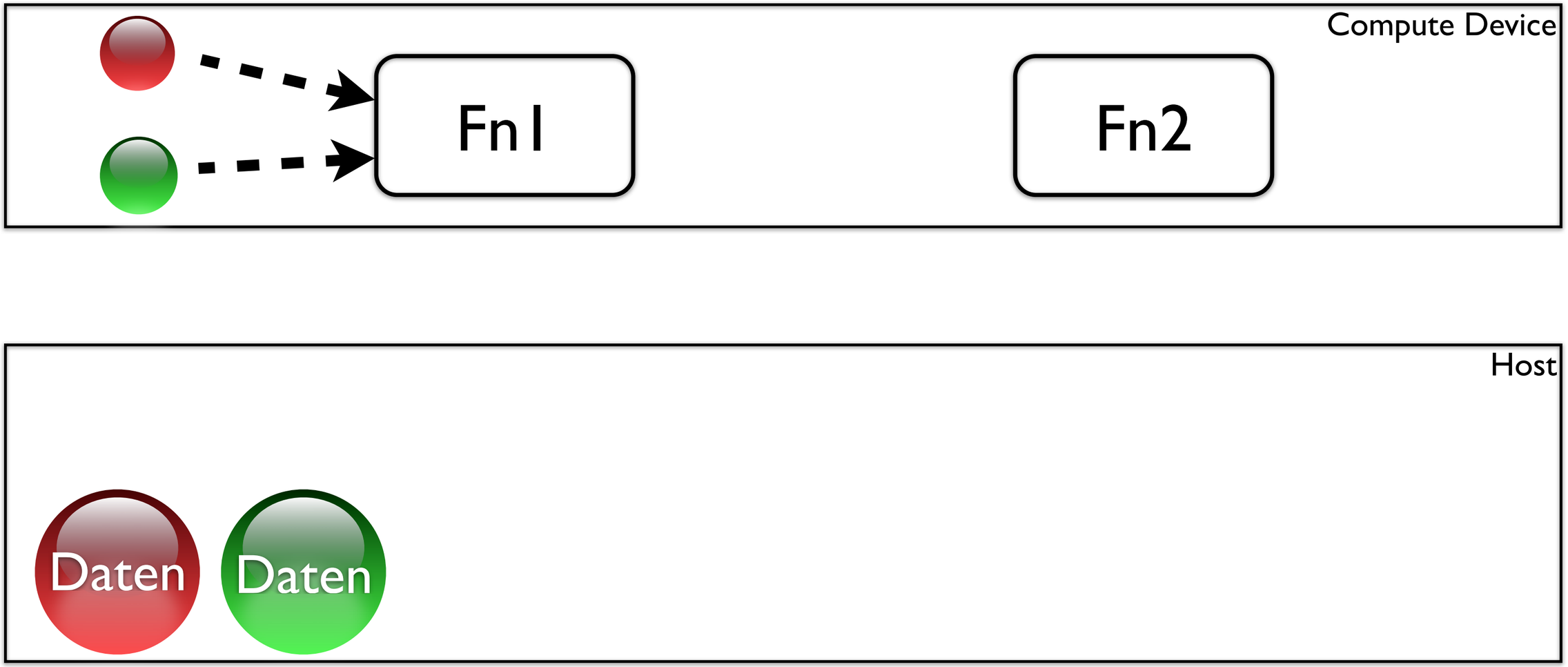
Speichermodell OpenCL



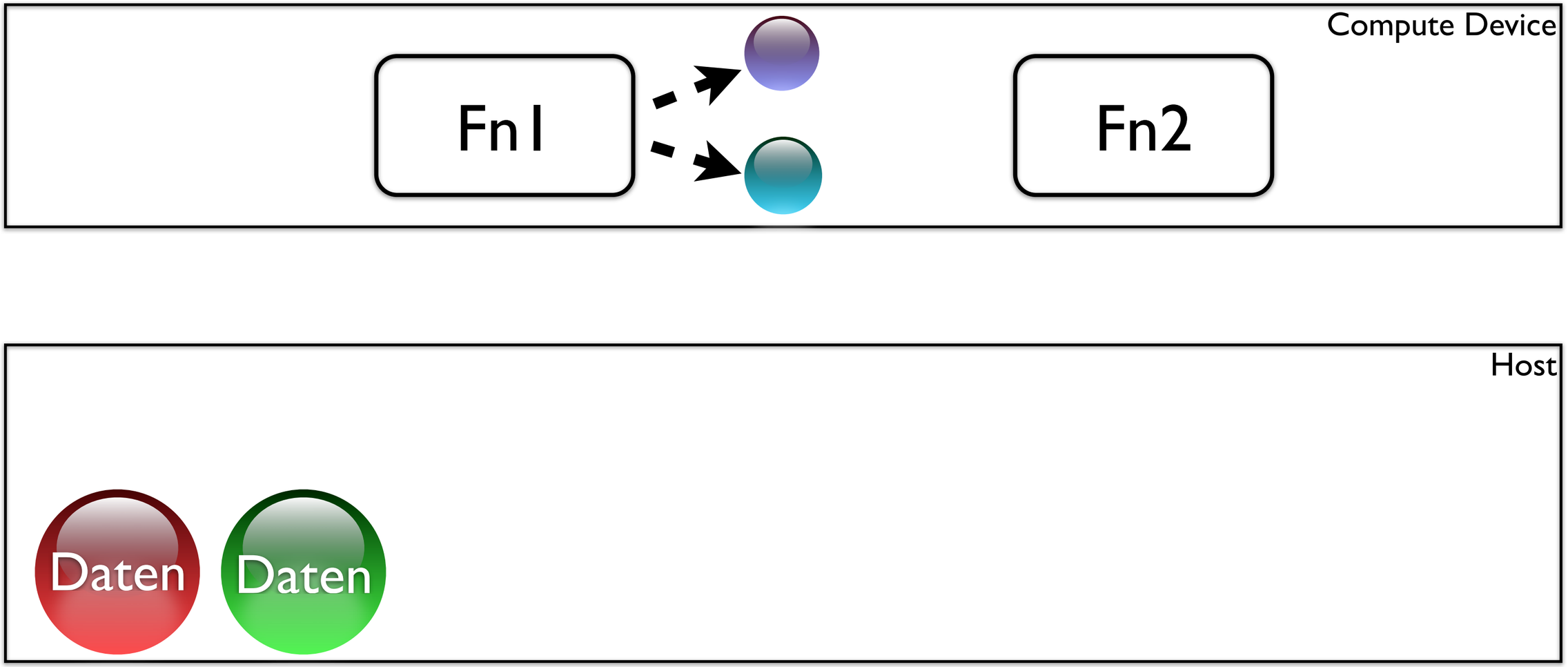
Speichermodell OpenCL



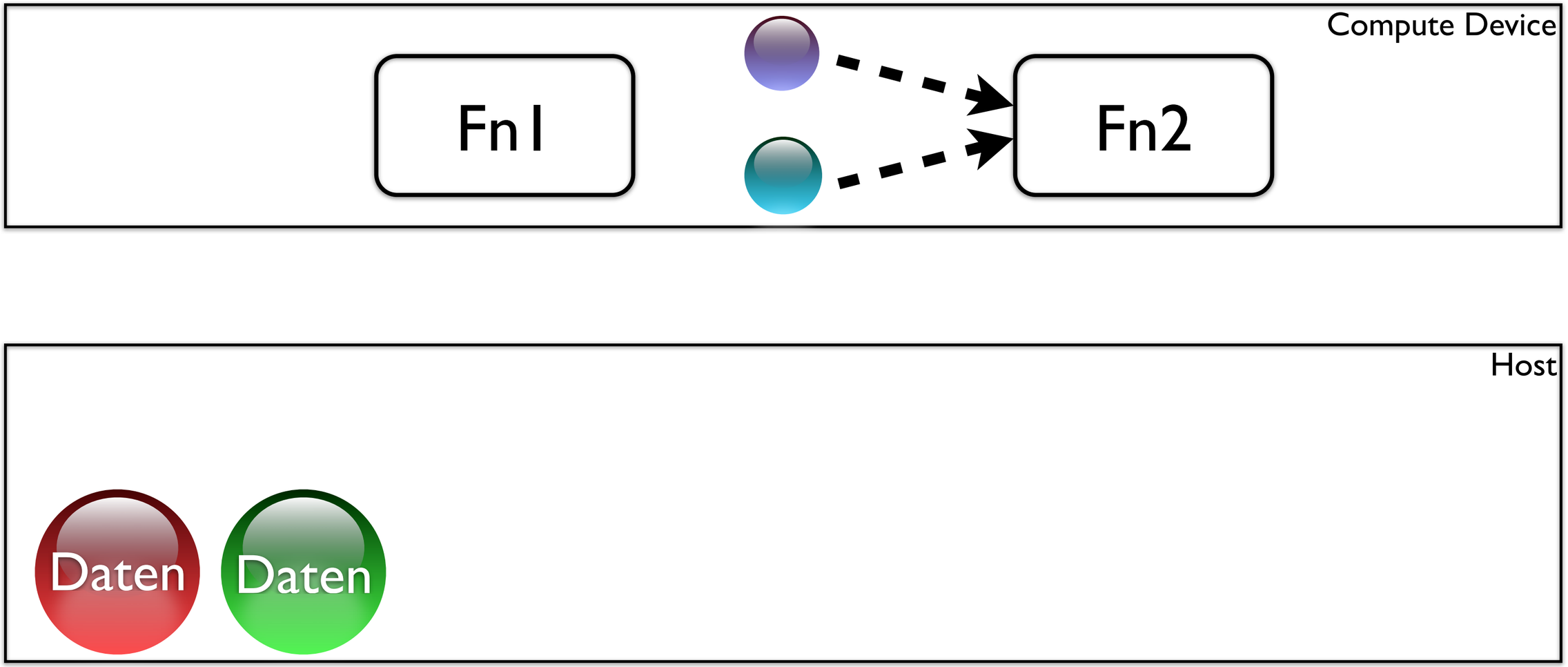
Speichermodell OpenCL



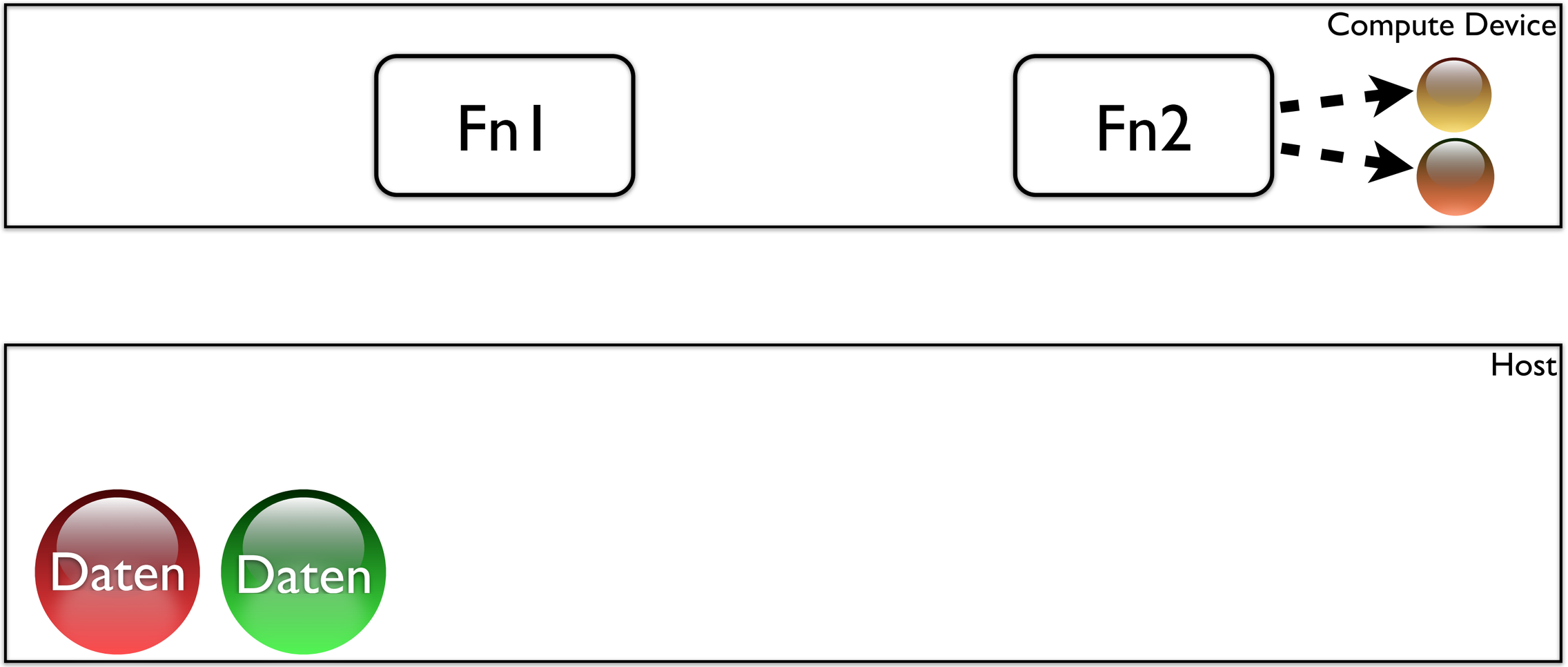
Speichermodell OpenCL



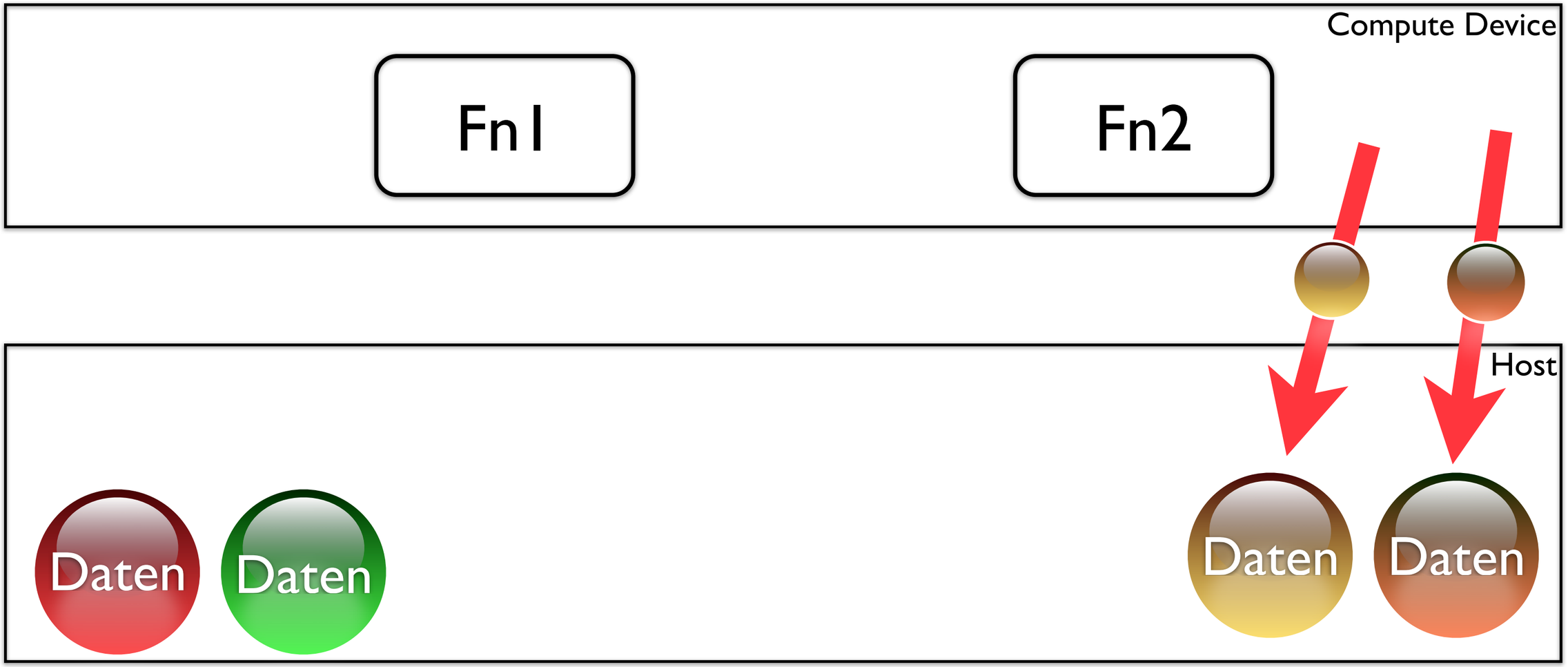
Speichermodell OpenCL



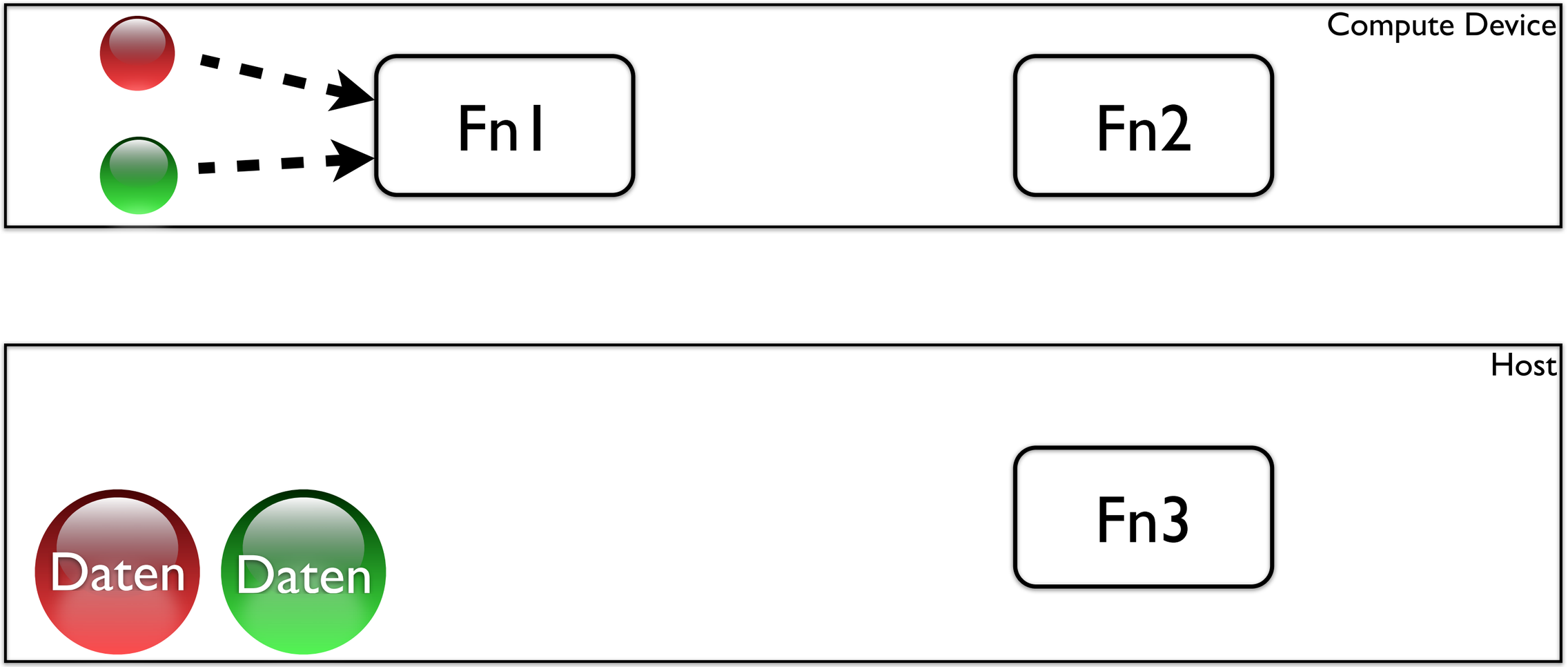
Speichermodell OpenCL



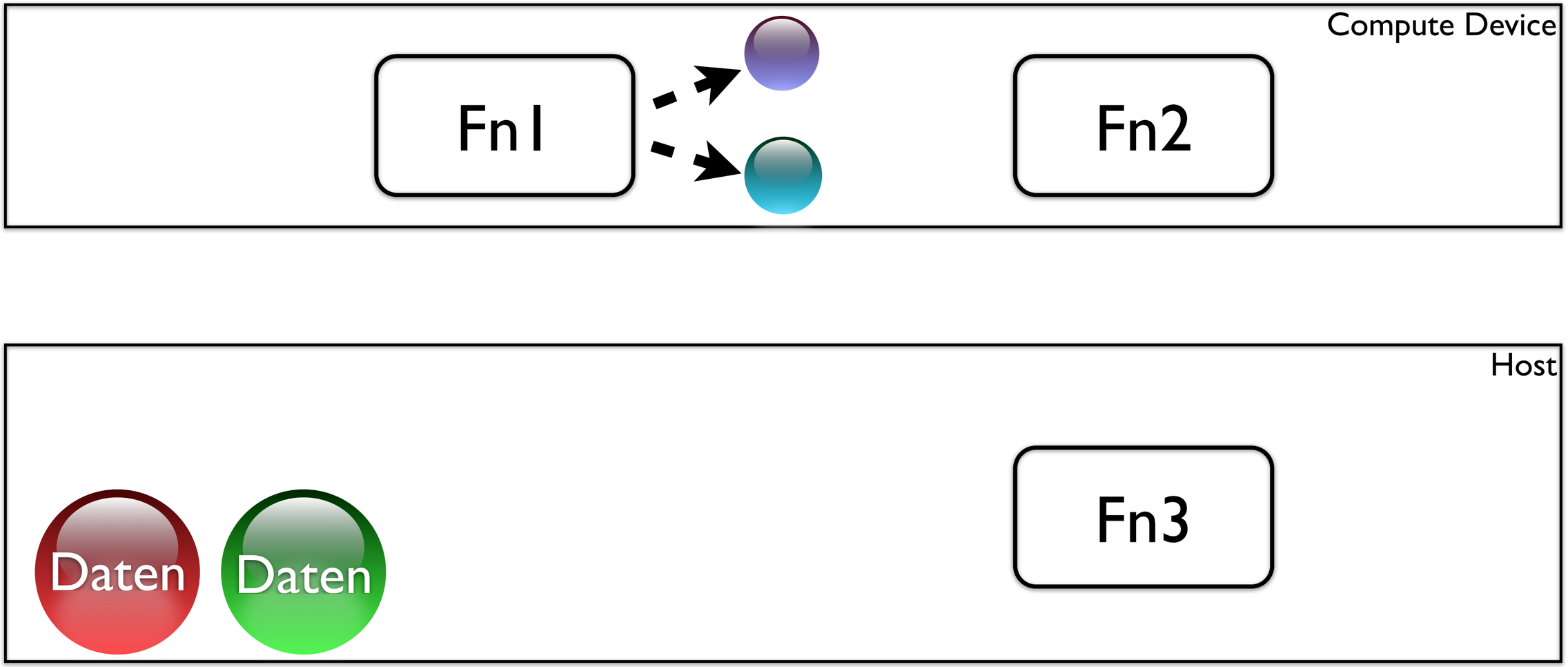
Speichermodell OpenCL



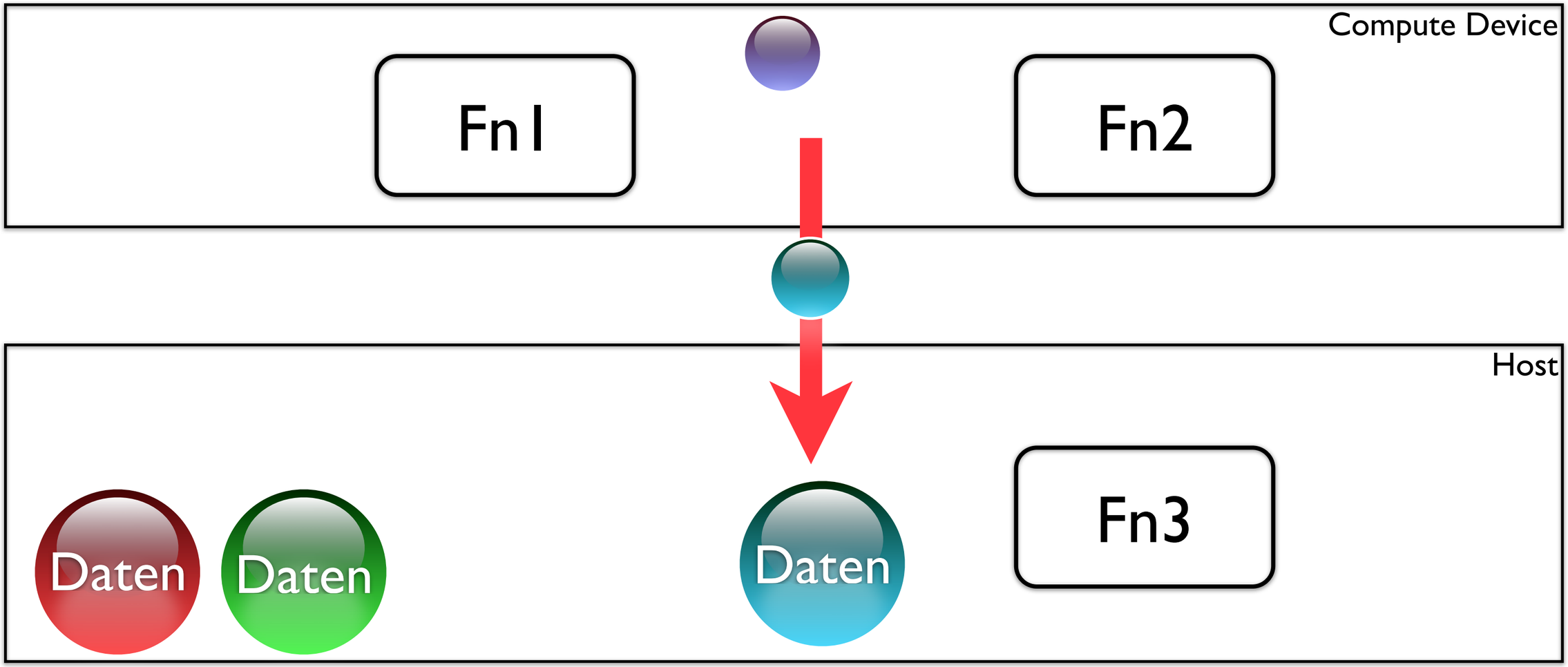
Speichermodell OpenCL



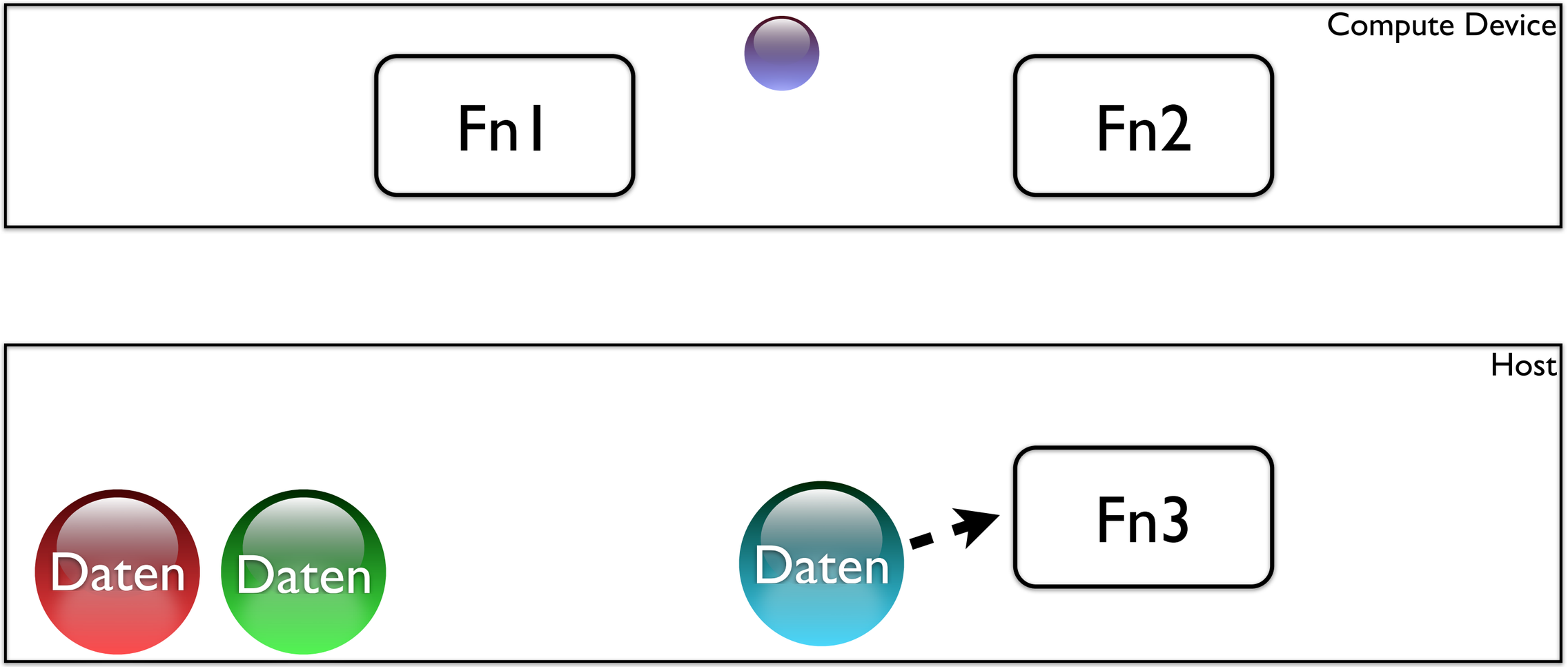
Speichermodell OpenCL



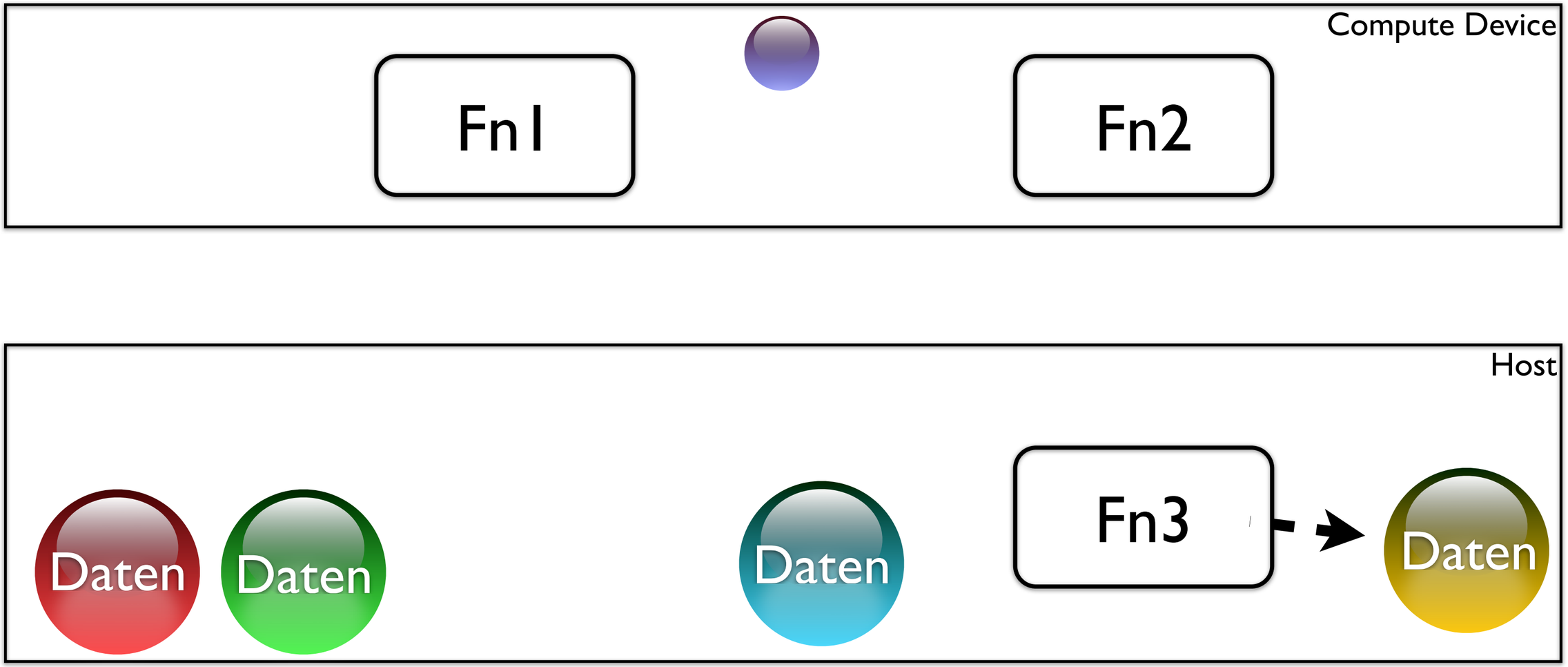
Speichermodell OpenCL



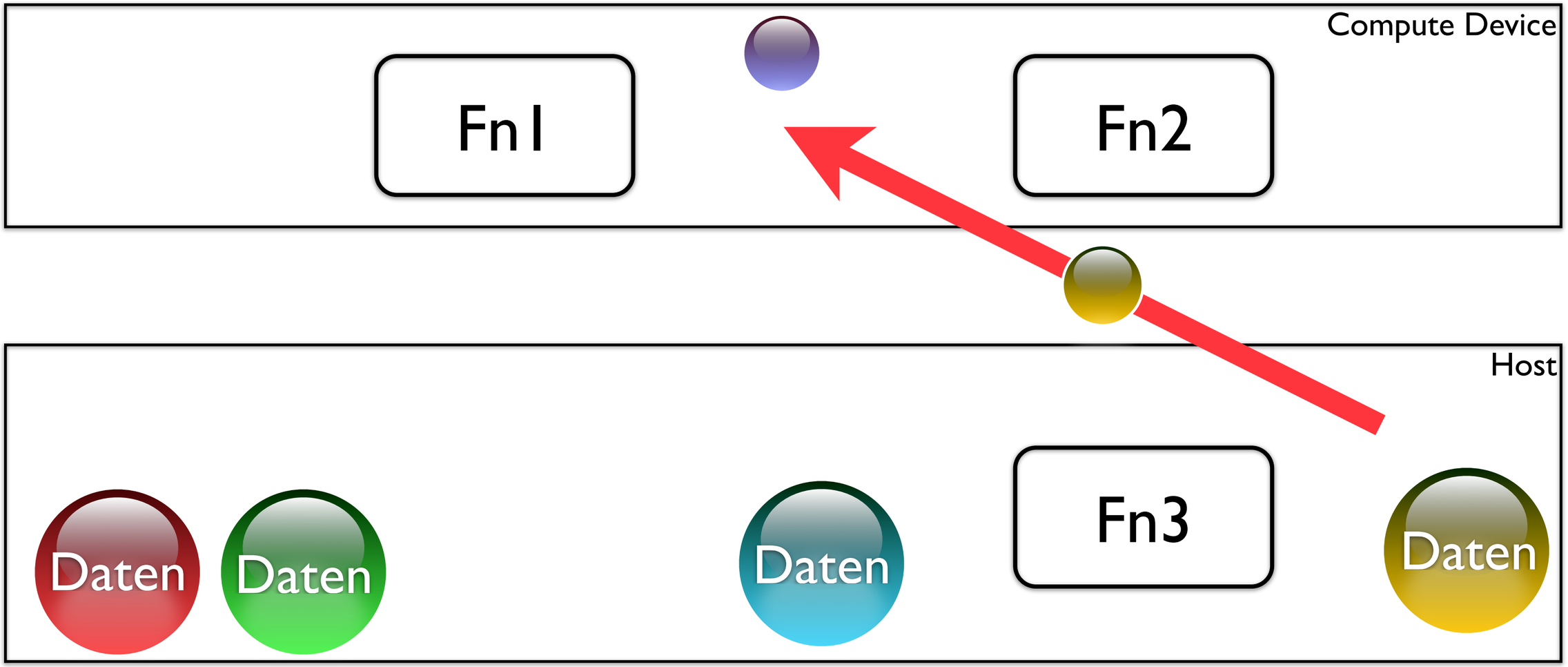
Speichermodell OpenCL



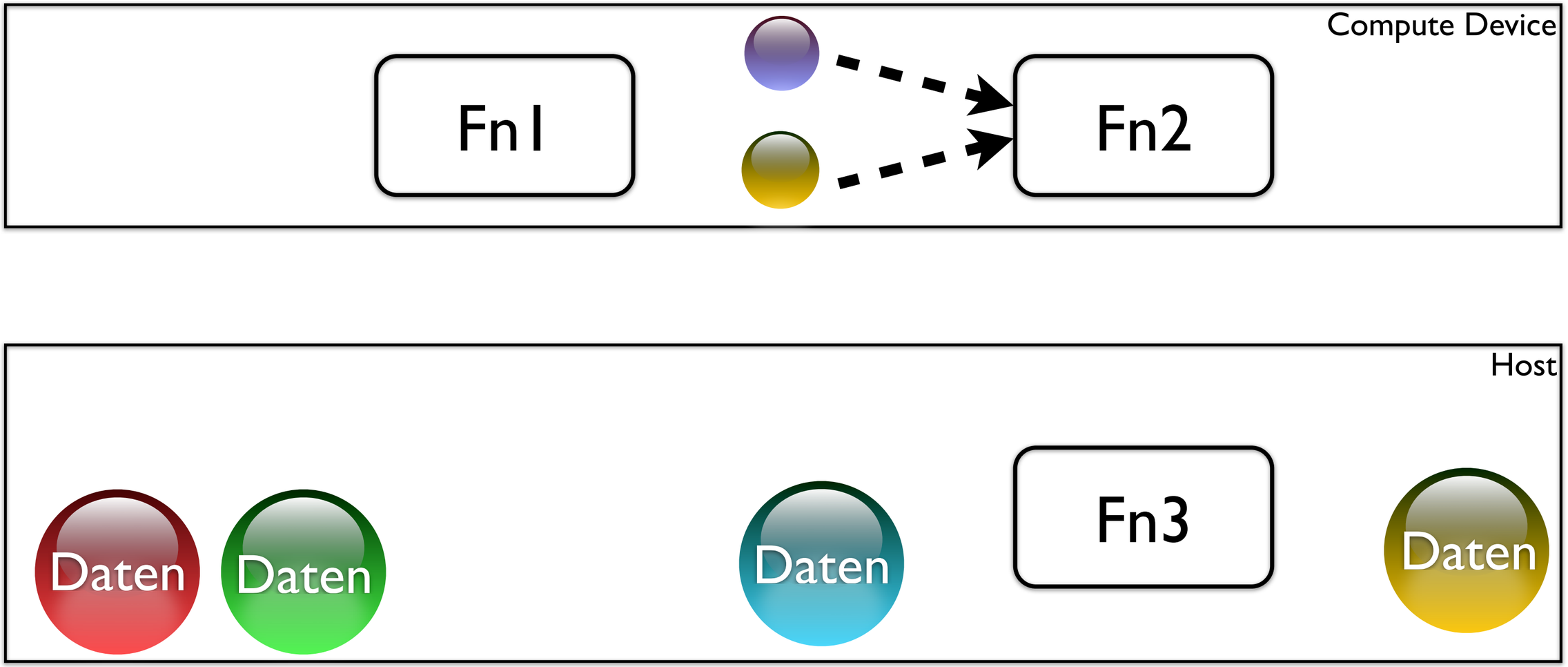
Speichermodell OpenCL



Speichermodell OpenCL



Speichermodell OpenCL



Demo

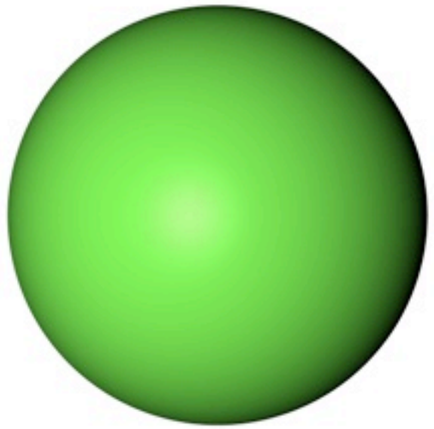
Hauptgericht

Think big

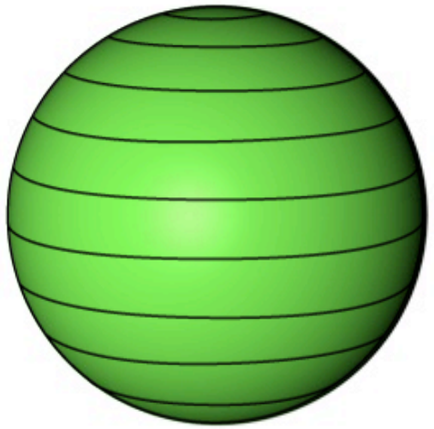
Think many

Think many:
So viele wie möglich
gleiche Operationen auf
verschiedene Objekte
(und zwar parallel!)

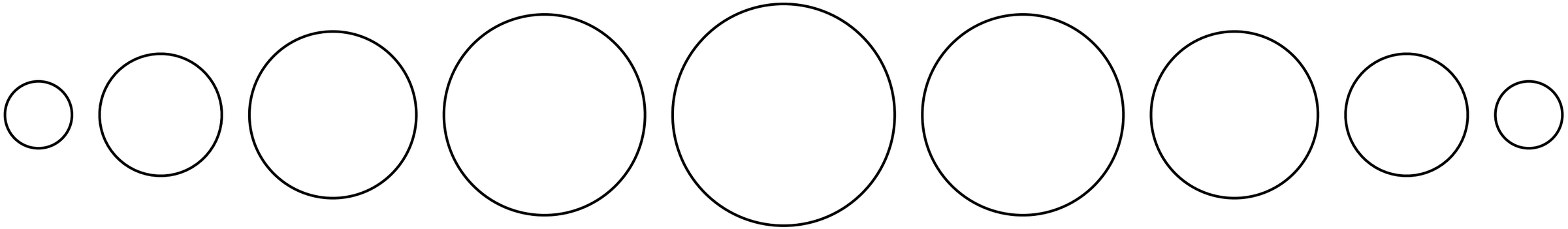
Der Slice-Algorithmus



Der Slice-Algorithmus



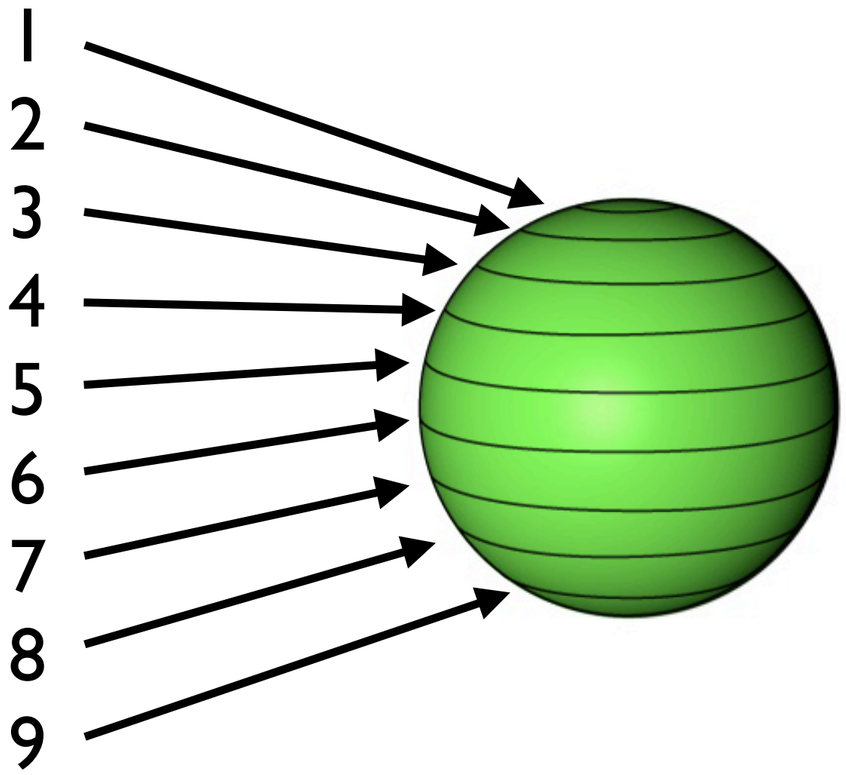
Der Slice-Algorithmus



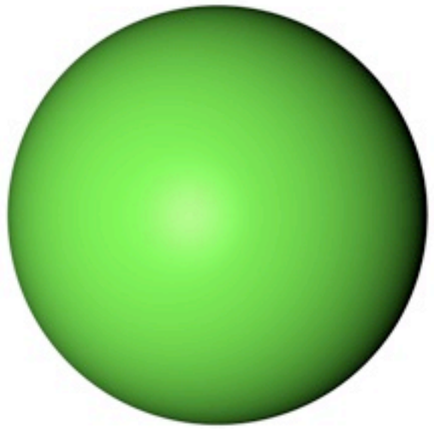
Der Slice-Algorithmus

- Index für Dreiecke anlegen
- *Für jede Schnittebene:*
  - Alle betroffenen Dreiecke schneiden
  - Umriss aus geschnittenen Kanten zusammensetzen

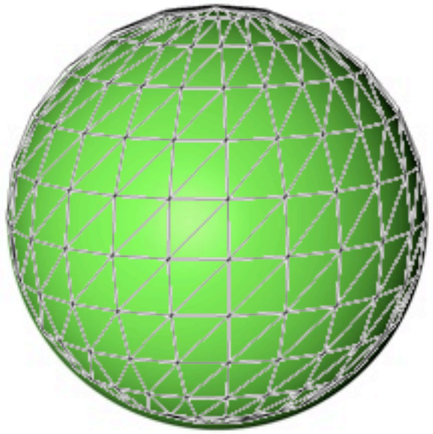
Der Slice-Algorithmus



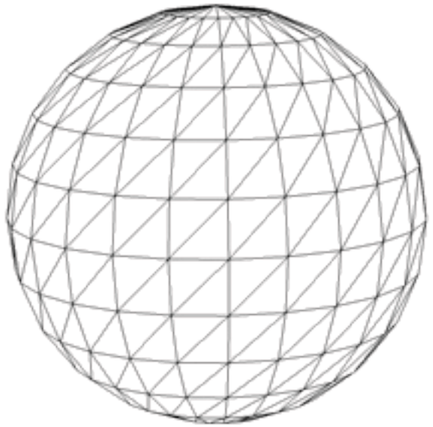
Der Slice-Algorithmus



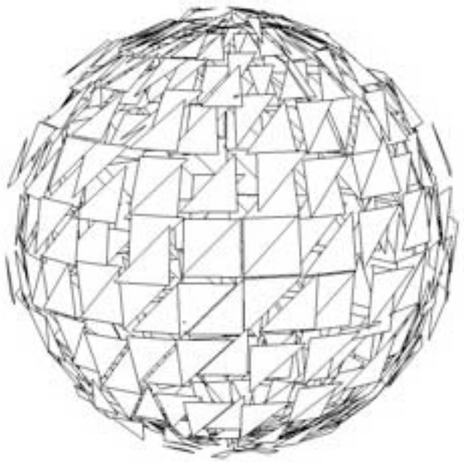
Der Slice-Algorithmus



Der Slice-Algorithmus

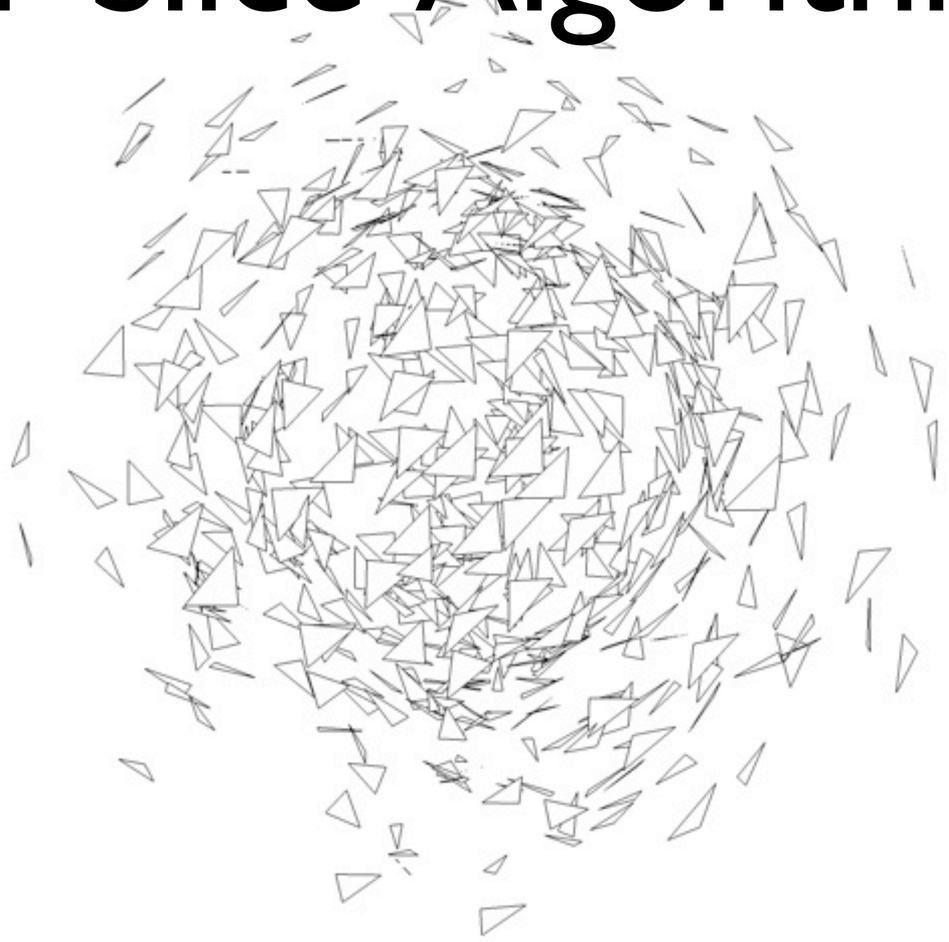


Der Slice-Algorithmus

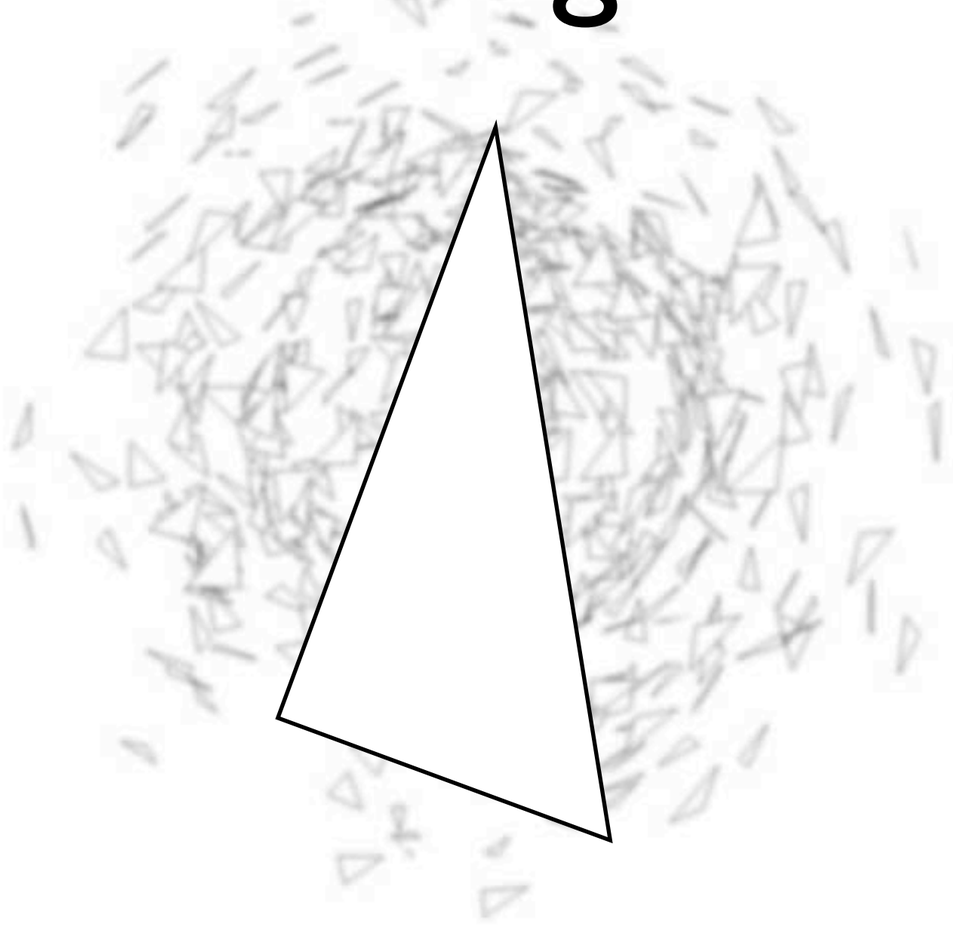


Der Slice-Algorithmus

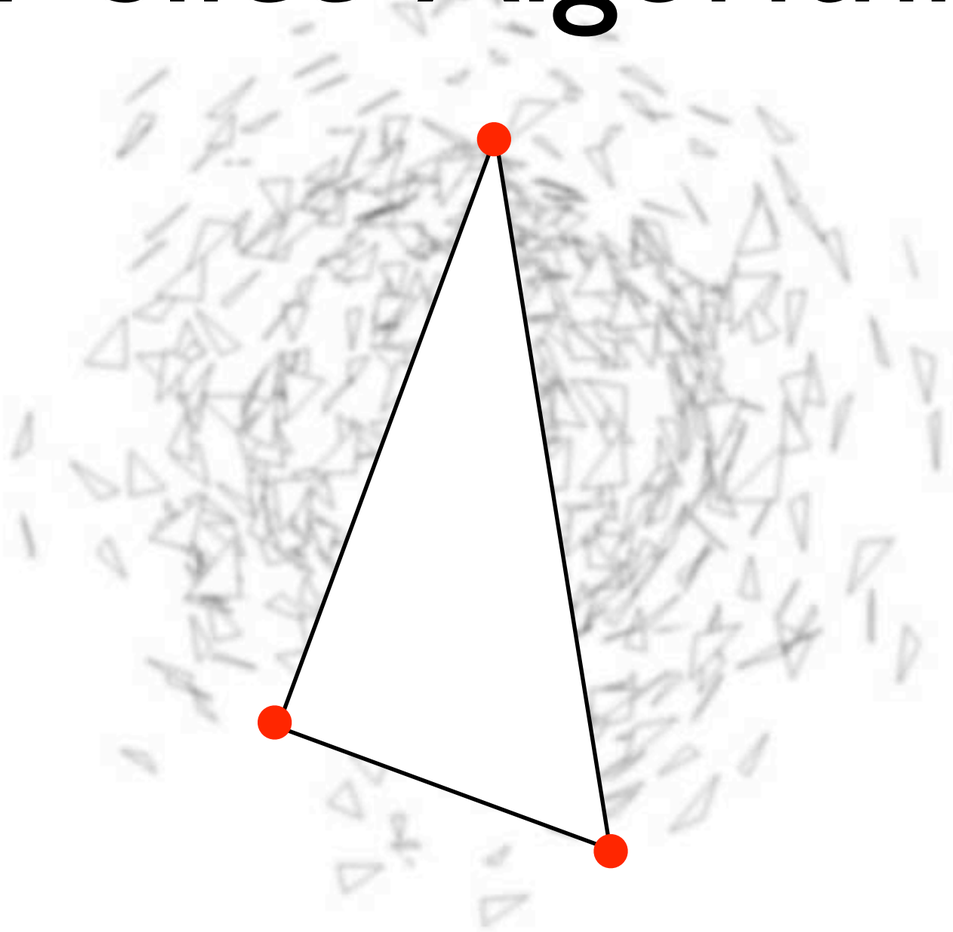
>> 9



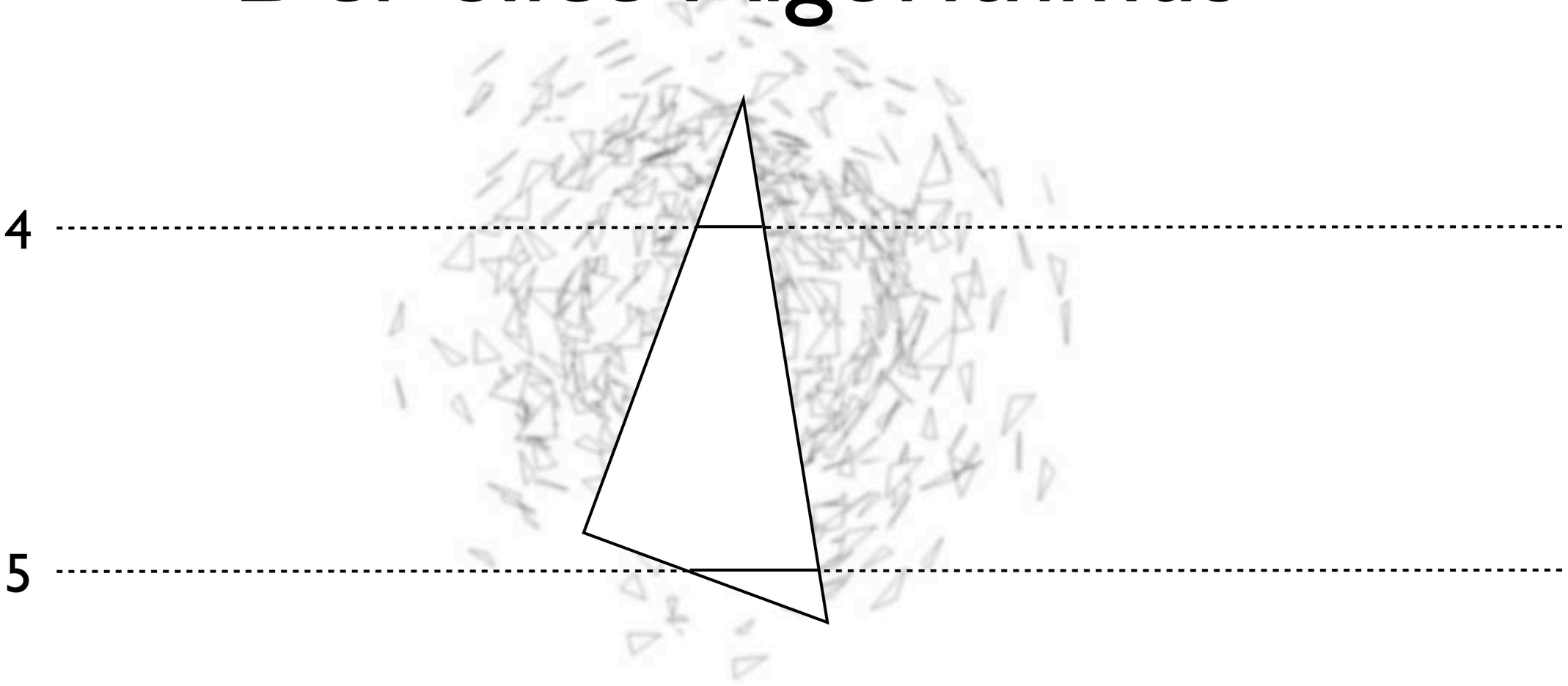
Der Slice-Algorithmus



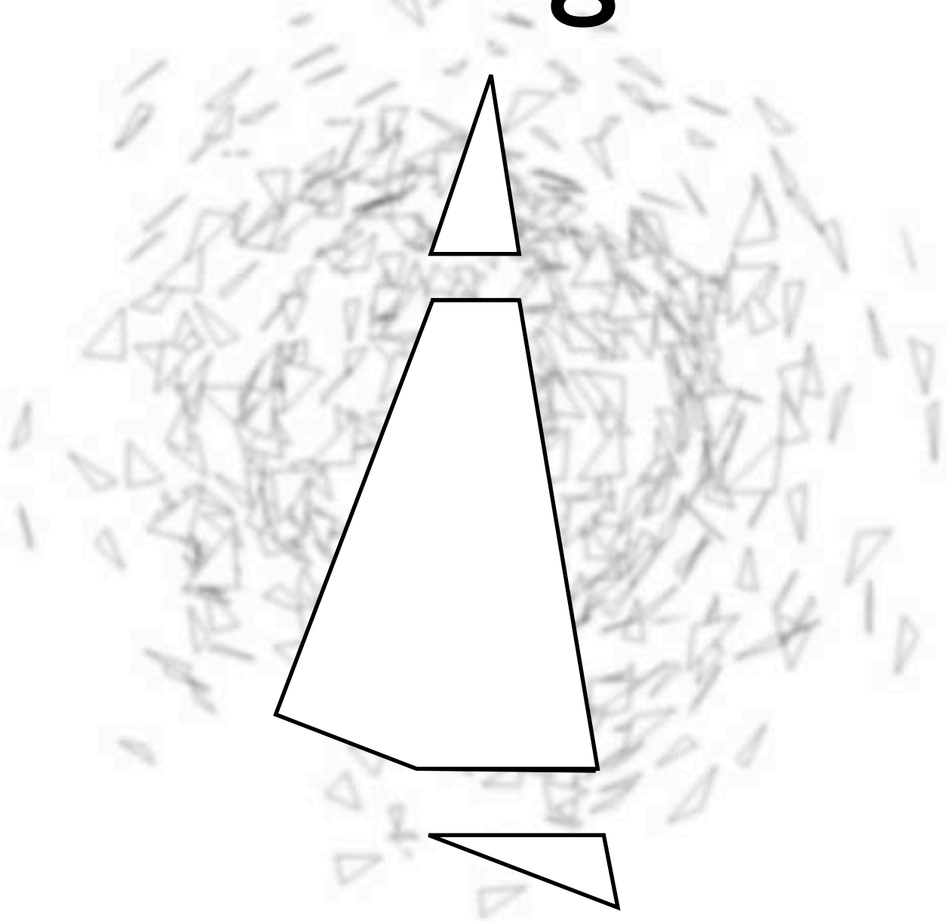
Der Slice-Algorithmus



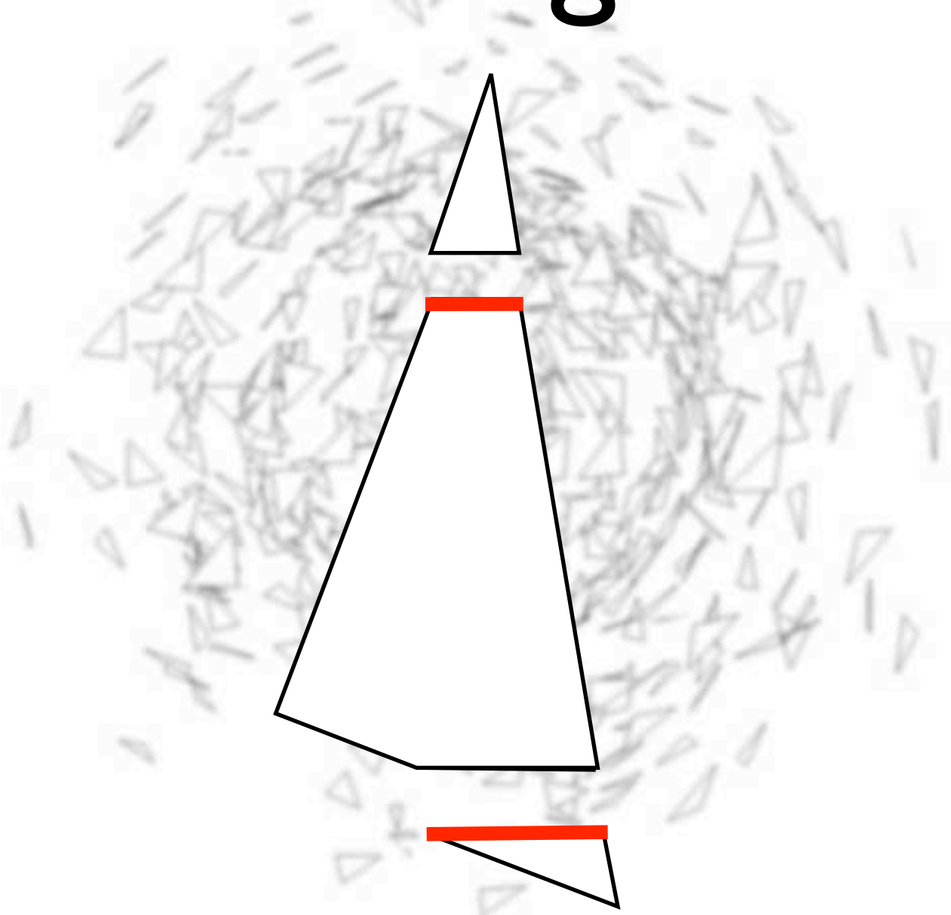
Der Slice-Algorithmus



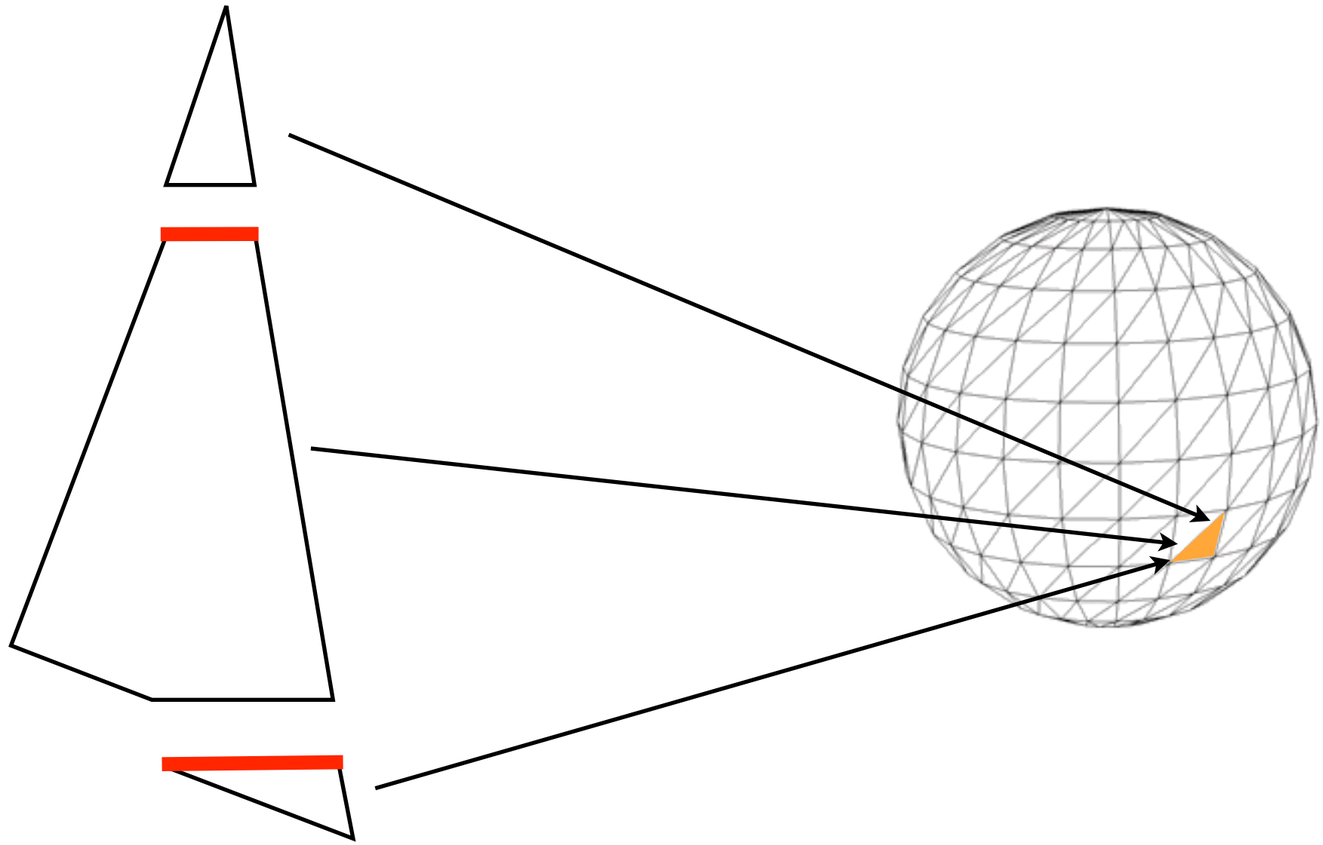
Der Slice-Algorithmus



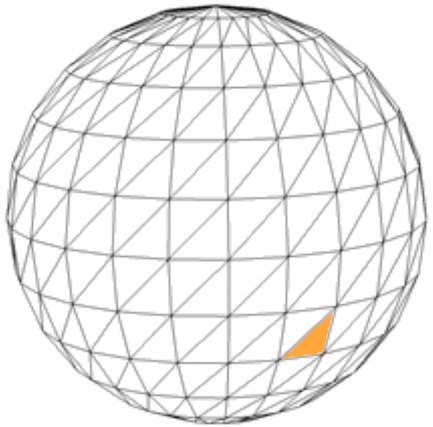
Der Slice-Algorithmus



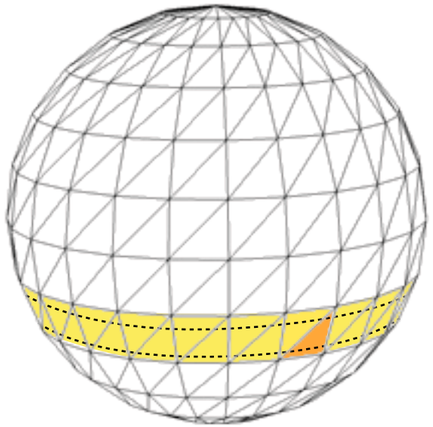
Der Slice-Algorithmus



Der Slice-Algorithmus



Der Slice-Algorithmus



Der Slice-Algorithmus

- Index für Dreiecke anlegen
- *Für jedes Dreieck:*
  - Dreieck schneiden
  - Schnittkante(n) speichern
- *Für alle gespeicherten Schnittkanten:*
  - Umrisse zusammensetzen

Der Slice-Algorithmus

- Index für Dreiecke anlegen
- *Für jede Schnittebene:*
  - Alle betroffenen Dreiecke schneiden
  - **Umriss aus geschnittenen Kanten zusammensetzen**

Der Slice-Algorithmus

- Index für Dreiecke anlegen
- *Für jedes Dreieck:*
  - Dreieck schneiden
  - Schnittkante(n) speichern
- *Für alle gespeicherten Schnittkanten:*
  - **Umrisse zusammensetzen**

Demo

Pleasant3D

Kostenloser Download & Link zu Sourcen:

http://pleasantsoftware.com/developer/pleasant3d/

Macoun'10