RenderScript Kernel Invocation Functions and Types

Overview

The rsForEach() function can be used to invoke the root kernel of a script.

The other functions are used to get the characteristics of the invocation of an executing kernel, like dimensions and current indices. These functions take a rs_kernel_context as argument.

Summary

Types	
rs_for_each_strategy_t	Suggested cell processing order
rs_kernel	Handle to a kernel function
rs_kernel_context	Handle to a kernel invocation context
rs_script_call_t	Cell iteration information

Functions	
rsForEach	Launches a kernel
rsForEachInternal	(Internal API) Launch a kernel in the current Script (with the slot number)
rsForEachWithOptions	Launches a kernel with options
rsGetArray0	Index in the Array0 dimension for the specified kernel context
rsGetArray1	Index in the Array1 dimension for the specified kernel context
rsGetArray2	Index in the Array2 dimension for the specified kernel context
rsGetArray3	Index in the Array3 dimension for the specified kernel context
rsGetDimArray0	Size of the Array0 dimension for the specified kernel context
rsGetDimArray1	Size of the Array1 dimension for the specified kernel context
rsGetDimArray2	Size of the Array2 dimension for the specified kernel context
rsGetDimArray3	Size of the Array3 dimension for the specified kernel context
rsGetDimHasFaces	Presence of more than one face for the specified kernel context
rsGetDimLod	Number of levels of detail for the specified kernel context
rsGetDimX	Size of the X dimension for the specified kernel context
rsGetDimY	Size of the Y dimension for the specified kernel context
rsGetDimZ	Size of the Z dimension for the specified kernel context
rsGetFace	Coordinate of the Face for the specified kernel context
rsGetLod	Index in the Levels of Detail dimension for the specified kernel context

Types

rs_for_each_strategy_t : Suggested cell processing order

rs_for_each_strategy_t : Suggested cell processing order

An enum with the following values:

RS_FOR_EACH_STRATEGY_SERIAL = 0	Prefer contiguous memory regions.
RS_FOR_EACH_STRATEGY_DONT_CARE = 1	No prefrences.
RS_FOR_EACH_STRATEGY_DST_LINEAR = 2	Prefer DST.
RS_FOR_EACH_STRATEGY_TILE_SMALL = 3	Prefer processing small rectangular regions.
RS_FOR_EACH_STRATEGY_TILE_MEDIUM = 4	Prefer processing medium rectangular regions.
RS_FOR_EACH_STRATEGY_TILE_LARGE = 5	Prefer processing large rectangular regions.

This type is used to suggest how the invoked kernel should iterate over the cells of the allocations. This is a hint only. Implementations may not follow the suggestion.

This specification can help the caching behavior of the running kernel, e.g. the cache locality when the processing is distributed over multiple cores.

rs_kernel : Handle to a kernel function

A typedef of: void* Added in API level 24

An opaque type for a function that is defined with the kernel attribute. A value of this type can be used in a rsForEach call to launch a kernel.

rs_kernel_context : Handle to a kernel invocation context

A typedef of: const struct rs_kernel_context_t * Added in API level 23

The kernel context contains common characteristics of the allocations being iterated over, like dimensions. It also contains rarely used indices of the currently processed cell, like the Array0 index or the current level of detail.

You can access the kernel context by adding a special parameter named "context" of type rs_kernel_context to your kernel function. See rsGetDimX() and rsGetArray0() for examples.

rs_script_call_t : Cell iteration information

A structure with the following fields:

rs_for_each_strategy_t strategy	Currently ignored. In the future, will be suggested cell iteration strategy.
uint32_t xStart	Starting index in the X dimension.
uint32_t xEnd	Ending index (exclusive) in the X dimension.
uint32_t yStart	Starting index in the Y dimension.
uint32_t yEnd	Ending index (exclusive) in the Y dimension.
uint32_t zStart	Starting index in the Z dimension.
uint32_t zEnd	Ending index (exclusive) in the Z dimension.
uint32_t arrayStart	Starting index in the Array0 dimension.
uint32_t arrayEnd	Ending index (exclusive) in the Array0 dimension.
uint32_t array1Start	Starting index in the Array1 dimension.
uint32_t array1End	Ending index (exclusive) in the Array1 dimension.
uint32_t array2Start	Starting index in the Array2 dimension.
uint32_t array2End	Ending index (exclusive) in the Array2 dimension.
uint32_t array3Start	Starting index in the Array3 dimension.
uint32_t array3End	Ending index (exclusive) in the Array3 dimension.

This structure is used to provide iteration information to a rsForEach call. It is currently used to restrict processing to a subset of cells. In future versions, it will also be used to provide hint on how to best iterate over the cells.

The Start fields are inclusive and the End fields are exclusive. E.g. to iterate over cells 4, 5, 6, and 7 in the X dimension, set xStart to 4 and xEnd to 8.

Functions

rsForEach : Launches a kernel

void rsForEach(rs_kernel kernel, ...);	Added in API level 24
void rsForEach(rs_script script, rs_allocation input, rs_allocation output);	API level 14 - 23
void rsForEach(rs_script script, rs_allocation input, rs_allocation output, const void* usrData);	Removed from API level 14 and higher
void rsForEach(rs_script script, rs_allocation input, rs_allocation output, const void* usrData, const rs_script_call_t* sc);	Removed from API level 14 and higher
void rsForEach(rs_script script, rs_allocation input, rs_allocation output, const void* usrData, size_t usrDataLen);	API level 14 - 20
void rsForEach(rs_script script, rs_allocation input, rs_allocation output, const void* usrData, size_t usrDataLen, const rs_script_call_t* sc);	API level 14 - 20

Parameters

script	Script to call.
input	Allocation to source data from.
output	Allocation to write date into.
usrData	User defined data to pass to the script. May be NULL.
sc	Extra control information used to select a sub-region of the allocation to be processed or suggest a walking strategy. May be NULL.
usrDataLen	Size of the userData structure. This will be used to perform a shallow copy of the data if necessary.
kernel	Function designator to a function that is defined with the kernel attribute.
...	Input and output allocations

Runs the kernel over zero or more input allocations. They are passed after the rs_kernel argument. If the specified kernel returns a value, an output allocation must be specified as the last argument. All input allocations, and the output allocation if it exists, must have the same dimensions.

This is a synchronous function. A call to this function only returns after all the work has completed for all cells of the input allocations. If the kernel function returns any value, the call waits until all results have been written to the output allocation.

Up to API level 23, the kernel is implicitly specified as the kernel named "root" in the specified script, and only a single input allocation can be used. Starting in API level 24, an arbitrary kernel function can be used, as specified by the kernel argument. The script argument is removed. The kernel must be defined in the current script. In addition, more than one input can be used.

E.g.
```
float __attribute__((kernel)) square(float a) {
  return a * a;
}

void compute(rs_allocation ain, rs_allocation aout) {
  rsForEach(square, ain, aout);
}
```

rsForEachInternal : (Internal API) Launch a kernel in the current Script (with the slot number)

void rsForEachInternal(int slot, rs_script_call_t* options, int hasOutput, int numInputs, rs_allocation* allocs);	Added in API level 24

Parameters

slot	
options	
hasOutput	Indicates whether the kernel generates output
numInputs	Number of input allocations
allocs	Input and output allocations

Internal API to launch a kernel.

rsForEachWithOptions : Launches a kernel with options

void rsForEachWithOptions(rs_kernel kernel, rs_script_call_t* options, ...);   Added in API level 24

Parameters

kernel	Function designator to a function that is defined with the kernel attribute.
options	Launch options
...	Input and output allocations

Launches kernel in a way similar to rsForEach. However, instead of processing all cells in the input, this function only processes cells in the subspace of the index space specified in options. With the index space explicitly specified by options, no input or output allocation is required for a kernel launch using this API. If allocations are passed in, they must match the number of arguments and return value expected by the kernel function. The output allocation is present if and only if the kernel has a non-void return value.

E.g.,
```
rs_script_call_t opts = {0};
opts.xStart = 0;
opts.xEnd = dimX;
opts.yStart = 0;
opts.yEnd = dimY / 2;
rsForEachWithOptions(foo, &opts, out, out);
```

rsGetArray0 : Index in the Array0 dimension for the specified kernel context

uint32_t rsGetArray0(rs_kernel_context context);   Added in API level 23

Returns the index in the Array0 dimension of the cell being processed, as specified by the supplied kernel context.

The kernel context contains common characteristics of the allocations being iterated over and rarely used indices, like the Array0 index.

You can access the kernel context by adding a special parameter named "context" of type rs_kernel_context to your kernel function. E.g.
```
short RS_KERNEL myKernel(short value, uint32_t x, rs_kernel_context context) {
  // The current index in the common x, y, z dimensions are accessed by
  // adding these variables as arguments. For the more rarely used indices
  // to the other dimensions, extract them from the kernel context:
  uint32_t index_a0 = rsGetArray0(context);
  //...
}
```

This function returns 0 if the Array0 dimension is not present.

rsGetArray1 : Index in the Array1 dimension for the specified kernel context

uint32_t rsGetArray1(rs_kernel_context context);   Added in API level 23

Returns the index in the Array1 dimension of the cell being processed, as specified by the supplied kernel context. See rsGetArray0() for an explanation of the context.

Returns 0 if the Array1 dimension is not present.

rsGetArray2 : Index in the Array2 dimension for the specified kernel context

uint32_t rsGetArray2(rs_kernel_context context);   Added in API level 23

Returns the index in the Array2 dimension of the cell being processed, as specified by the supplied kernel context. See rsGetArray0() for an explanation of the context.

Returns 0 if the Array2 dimension is not present.

rsGetArray3 : Index in the Array3 dimension for the specified kernel context

uint32_t rsGetArray3(rs_kernel_context context);   Added in API level 23

Returns the index in the Array3 dimension of the cell being processed, as specified by the supplied kernel context. See rsGetArray0() for an explanation of the context.

Returns 0 if the Array3 dimension is not present.

rsGetDimArray0 : Size of the Array0 dimension for the specified kernel context

uint32_t rsGetDimArray0(rs_kernel_context context);   Added in API level 23

Returns the size of the Array0 dimension for the specified kernel context. See rsGetDimX() for an explanation of the context.

Returns 0 if the Array0 dimension is not present.

rsGetDimArray1 : Size of the Array1 dimension for the specified kernel context

uint32_t rsGetDimArray1(rs_kernel_context context);   Added in API level 23

Returns the size of the Array1 dimension for the specified kernel context. See rsGetDimX() for an explanation of the context.

Returns 0 if the Array1 dimension is not present.

rsGetDimArray2 : Size of the Array2 dimension for the specified kernel context

uint32_t rsGetDimArray2(rs_kernel_context context);   Added in API level 23

Returns the size of the Array2 dimension for the specified kernel context. See rsGetDimX() for an explanation of the context.

Returns 0 if the Array2 dimension is not present.

rsGetDimArray3 : Size of the Array3 dimension for the specified kernel context

uint32_t rsGetDimArray3(rs_kernel_context context);   Added in API level 23

Returns the size of the Array3 dimension for the specified kernel context. See rsGetDimX() for an explanation of the context.

Returns 0 if the Array3 dimension is not present.

rsGetDimHasFaces : Presence of more than one face for the specified kernel context

bool rsGetDimHasFaces(rs_kernel_context context);   Added in API level 23

Returns

Returns true if more than one face is present, false otherwise.

If the kernel is iterating over a cubemap, this function returns true if there's more than one face present. In all other cases, it returns false. See rsGetDimX() for an explanation of the context.

rsAllocationGetDimFaces() is similar but returns 0 or 1 instead of a bool.

rsGetDimLod : Number of levels of detail for the specified kernel context

uint32_t rsGetDimLod(rs_kernel_context context);   Added in API level 23

Returns the number of levels of detail for the specified kernel context. This is useful for mipmaps. See rsGetDimX() for an explanation of the context.

Returns 0 if Level of Detail is not used.

rsAllocationGetDimLOD() is similar but returns 0 or 1 instead the actual number of levels.

rsGetDimX : Size of the X dimension for the specified kernel context

uint32_t rsGetDimX(rs_kernel_context context);   Added in API level 23

Returns the size of the X dimension for the specified kernel context.

The kernel context contains common characteristics of the allocations being iterated over and rarely used indices, like the Array0 index.

You can access it by adding a special parameter named "context" of type rs_kernel_context to your kernel function. E.g.

```
int4 RS_KERNEL myKernel(int4 value, rs_kernel_context context) {
  uint32_t size = rsGetDimX(context); //...
```

To get the dimension of specific allocation, use rsAllocationGetDimX().

rsGetDimY : Size of the Y dimension for the specified kernel context

uint32_t rsGetDimY(rs_kernel_context context);   Added in API level 23

Returns the size of the X dimension for the specified kernel context. See rsGetDimX() for an explanation of the context.

Returns 0 if the Y dimension is not present.

To get the dimension of specific allocation, use rsAllocationGetDimY().

rsGetDimZ : Size of the Z dimension for the specified kernel context

uint32_t rsGetDimZ(rs_kernel_context context);   Added in API level 23

Returns the size of the Z dimension for the specified kernel context. See rsGetDimX() for an explanation of the context.

Returns 0 if the Z dimension is not present.

To get the dimension of specific allocation, use rsAllocationGetDimZ().

rsGetFace : Coordinate of the Face for the specified kernel context

rs_allocation_cubemap_face rsGetFace(rs_kernel_context context);   Added in API level 23

Returns the face on which the cell being processed is found, as specified by the supplied kernel context. See rsGetArray0() for an explanation of the context.

Returns RS_ALLOCATION_CUBEMAP_FACE_POSITIVE_X if the face dimension is not present.

rsGetLod : Index in the Levels of Detail dimension for the specified kernel context

uint32_t rsGetLod(rs_kernel_context context);   Added in API level 23

Returns the index in the Levels of Detail dimension of the cell being processed, as specified by the supplied kernel context. See rsGetArray0() for an explanation of the context.

Returns 0 if the Levels of Detail dimension is not present.