RenderScript Object Characteristics Functions

Overview

The functions below can be used to query the characteristics of an Allocation, Element, or Sampler object. These objects are created from Java. You can't create them from a script.

Allocations:

Allocations are the primary method used to pass data to and from RenderScript kernels.

They are a structured collection of cells that can be used to store bitmaps, textures, arbitrary data points, etc.

This collection of cells may have many dimensions (X, Y, Z, Array0, Array1, Array2, Array3), faces (for cubemaps), and level of details (for mipmapping).

See the android.renderscript.Allocation for details on to create Allocations.

Elements:

The term "element" is used a bit ambiguously in RenderScript, as both type information for the cells of an Allocation and the instantiation of that type. For example:

- rs_element is a handle to a type specification, and
- In functions like rsGetElementAt(), "element" means the instantiation of the type, i.e. a cell of an Allocation.

The functions below let you query the characteristics of the type specificiation.

An Element can specify a simple data types as found in C, e.g. an integer, float, or boolean. It can also specify a handle to a RenderScript object. See rs_data_type for a list of basic types.

Elements can specify fixed size vector (of size 2, 3, or 4) versions of the basic types. Elements can be grouped together into complex Elements, creating the equivalent of C structure definitions.

Elements can also have a kind, which is semantic information used to interpret pixel data. See rs_data_kind.

When creating Allocations of common elements, you can simply use one of the many predefined Elements like F32_2.

To create complex Elements, use the Element.Builder Java class.

Samplers:

Samplers objects define how Allocations can be read as structure within a kernel. See android.renderscript.S.

Summary

Functions	
rsAllocationGetDimFaces	Presence of more than one face
rsAllocationGetDimLOD	Presence of levels of detail
rsAllocationGetDimX	Size of the X dimension
rsAllocationGetDimY	Size of the Y dimension
rsAllocationGetDimZ	Size of the Z dimension

rsAllocationGetElement	Get the object that describes the cell of an Allocation
rsClearObject	Release an object
rsElementGetBytesSize	Size of an Element
rsElementGetDataKind	Kind of an Element
rsElementGetDataType	Data type of an Element
rsElementGetSubElement	Sub-element of a complex Element
rsElementGetSubElementArraySize	Array size of a sub-element of a complex Element
rsElementGetSubElementCount	Number of sub-elements
rsElementGetSubElementName	Name of a sub-element
rsElementGetSubElementNameLength	Length of the name of a sub-element
rsElementGetSubElementOffsetBytes	Offset of the instantiated sub-element
rsElementGetVectorSize	Vector size of the Element
rsIsObject	Check for an empty handle
rsSamplerGetAnisotropy	Anisotropy of the Sampler
rsSamplerGetMagnification	Sampler magnification value
rsSamplerGetMinification	Sampler minification value
rsSamplerGetWrapS	Sampler wrap S value
rsSamplerGetWrapT	Sampler wrap T value

Deprecated Functions	
rsGetAllocation	**Deprecated**. Return the Allocation for a given pointer

Functions

rsAllocationGetDimFaces : Presence of more than one face

uint32_t rsAllocationGetDimFaces(rs_allocation a);

Returns

Returns 1 if more than one face is present, 0 otherwise.

If the Allocation is a cubemap, this function returns 1 if there's more than one face present. In all other cases, it returns 0.

Use rsGetDimHasFaces() to get the dimension of a currently running kernel.

rsAllocationGetDimLOD : Presence of levels of detail

uint32_t rsAllocationGetDimLOD(rs_allocation a);

Returns

Returns 1 if more than one LOD is present, 0 otherwise.

Query an Allocation for the presence of more than one Level Of Detail. This is useful for mipmaps.

Use rsGetDimLod() to get the dimension of a currently running kernel.

rsAllocationGetDimX : Size of the X dimension

uint32_t rsAllocationGetDimX(rs_allocation a);

Returns

X dimension of the Allocation.

Returns the size of the X dimension of the Allocation.

Use rsGetDimX() to get the dimension of a currently running kernel.

rsAllocationGetDimY : Size of the Y dimension

uint32_t rsAllocationGetDimY(rs_allocation a);

Returns

Y dimension of the Allocation.

Returns the size of the Y dimension of the Allocation. If the Allocation has less than two dimensions, returns 0.

Use rsGetDimY() to get the dimension of a currently running kernel.

rsAllocationGetDimZ : Size of the Z dimension

uint32_t rsAllocationGetDimZ(rs_allocation a);

Returns

Z dimension of the Allocation.

Returns the size of the Z dimension of the Allocation. If the Allocation has less than three dimensions, returns 0.

Use rsGetDimZ() to get the dimension of a currently running kernel.

rsAllocationGetElement : Get the object that describes the cell of an Allocation

rs_element rsAllocationGetElement(rs_allocation a);

Parameters

a Allocation to get data from.

Returns

Element describing Allocation layout.

Get the Element object describing the type, kind, and other characteristics of a cell of an Allocation. See the rsElement* functions below.

rsClearObject : Release an object

void rsClearObject(rs_allocation* dst);	
void rsClearObject(rs_element* dst);	
void rsClearObject(rs_font* dst);	When compiling for 32 bits. Removed from API level 23 and higher
void rsClearObject(rs_mesh* dst);	When compiling for 32 bits. Removed from API level 23 and higher
void rsClearObject(rs_program_fragment* dst);	When compiling for 32 bits. Removed from API level 23 and higher
void rsClearObject(rs_program_raster* dst);	When compiling for 32 bits. Removed from API level 23 and higher
void rsClearObject(rs_program_store* dst);	When compiling for 32 bits. Removed from API level 23 and higher
void rsClearObject(rs_program_vertex* dst);	When compiling for 32 bits. Removed from API level 23 and higher
void rsClearObject(rs_sampler* dst);	
void rsClearObject(rs_script* dst);	
void rsClearObject(rs_type* dst);	

Tells the run time that this handle will no longer be used to access the the related object. If this was the last handle to that object, resource recovery may happen.

After calling this function, *dst will be set to an empty handle. See rsIsObject().

rsElementGetBytesSize : Size of an Element

uint32_t rsElementGetBytesSize(rs_element e); Added in API level 16

Returns the size in bytes that an instantiation of this Element will occupy.

rsElementGetDataKind : Kind of an Element

rs_data_kind rsElementGetDataKind(rs_element e); Added in API level 16

Returns the Element's data kind. This is used to interpret pixel data.

See rs_data_kind.

rsElementGetDataType : Data type of an Element

rs_data_type rsElementGetDataType(rs_element e); Added in API level 16

Returns the Element's base data type. This can be a type similar to C/C++ (e.g. RS_TYPE_UNSIGNED_8), a handle (e.g. RS_TYPE_ALLOCATION and RS_TYPE_ELEMENT), or a more complex numerical type (e.g. RS_TYPE_UNSIGNED_5_6_5 and RS_TYPE_MATRIX_4X4). See rs_data_type.

If the Element describes a vector, this function returns the data type of one of its items. Use rsElementGetVectorSize to get the size of the vector.

If the Element describes a structure, RS_TYPE_NONE is returned. Use the rsElementGetSub* functions to explore this complex Element.

rsElementGetSubElement : Sub-element of a complex Element

rs_element rsElementGetSubElement(rs_element e, uint32_t index); Added in API level 16

Parameters

e Element to query.

index Index of the sub-element to return.

Returns

Sub-element at the given index.

For Elements that represents a structure, this function returns the sub-element at the specified index.

If the Element is not a structure or the index is greater or equal to the number of sub-elements, an invalid handle is returned.

rsElementGetSubElementArraySize : Array size of a sub-element of a complex Element

uint32_t rsElementGetSubElementArraySize(rs_element e, uint32_t index); Added in API level 16

Parameters

e Element to query.

index Index of the sub-element.

Returns

Array size of the sub-element.

For complex Elements, sub-elements can be statically sized arrays. This function returns the array size of the sub-element at the index. This sub-element repetition is different than fixed size vectors.

rsElementGetSubElementCount : Number of sub-elements

uint32_t rsElementGetSubElementCount(rs_element e); Added in API level 16

Parameters

e Element to get data from.

Returns

Number of sub-elements.

Elements can be simple, such as an int or a float, or a structure with multiple sub-elements. This function returns zero for simple Elements and the number of sub-elements for complex Elements.

rsElementGetSubElementName : Name of a sub-element

uint32_t rsElementGetSubElementName(rs_element e, uint32_t index, char* name, uint32_t nameLength); Added in API level 16

Parameters

e	Element to get data from.
index	Index of the sub-element.
name	Address of the array to store the name into.
nameLength	Length of the provided name array.

Returns

Number of characters copied, excluding the null terminator.

For complex Elements, this function returns the name of the sub-element at the specified index.

rsElementGetSubElementNameLength : Length of the name of a sub-element

uint32_t rsElementGetSubElementNameLength(rs_element e, uint32_t index); Added in API level 16

Parameters

e	Element to get data from.
index	Index of the sub-element.

Returns

Length of the sub-element name including the null terminator.

For complex Elements, this function returns the length of the name of the sub-element at the specified index.

rsElementGetSubElementOffsetBytes : Offset of the instantiated sub-element

uint32_t rsElementGetSubElementOffsetBytes(rs_element e, uint32_t index); Added in API level 16

Parameters

e	Element to get data from.
index	Index of the sub-element.

Returns

Offset in bytes.

This function returns the relative position of the instantiation of the specified sub-element within the instantiation of the Element.

For example, if the Element describes a 32 bit float followed by a 32 bit integer, the offset return for the first will be 0 and the second 4.

rsElementGetVectorSize : Vector size of the Element

uint32_t rsElementGetVectorSize(rs_element e); Added in API level 16

Parameters

e	Element to get data from.

Returns

Length of the element vector.

Returns the Element's vector size. If the Element does not represent a vector, 1 is returned.

rsGetAllocation : Return the Allocation for a given pointer

rs_allocation rsGetAllocation(const void* p);

Deprecated. This function is deprecated and will be removed from the SDK in a future release.

Returns the Allocation for a given pointer. The pointer should point within a valid allocation. The results are undefined if the pointer is not from a valid Allocation.

rsIsObject : Check for an empty handle

bool rsIsObject(rs_allocation v);
bool rsIsObject(rs_element v);
bool rsIsObject(rs_font v); When compiling for 32 bits. Removed from API level 23 and higher
bool rsIsObject(rs_mesh v); When compiling for 32 bits. Removed from API level 23 and higher
bool rsIsObject(rs_program_fragment v); When compiling for 32 bits. Removed from API level 23 and higher
bool rsIsObject(rs_program_raster v); When compiling for 32 bits. Removed from API level 23 and higher
bool rsIsObject(rs_program_store v); When compiling for 32 bits. Removed from API level 23 and higher
bool rsIsObject(rs_program_vertex v); When compiling for 32 bits. Removed from API level 23 and higher
bool rsIsObject(rs_sampler v);
bool rsIsObject(rs_script v);
bool rsIsObject(rs_type v);

Returns true if the handle contains a non-null reference.

This function does not validate that the internal pointer used in the handle points to an actual valid object; it only checks for null.

This function can be used to check the Element returned by rsElementGetSubElement() or see if rsClearObject() has been called on a handle.

rsSamplerGetAnisotropy : Anisotropy of the Sampler

float rsSamplerGetAnisotropy(rs_sampler s); Added in API level 16

Get the Sampler's anisotropy.

See android.renderscript.S.

rsSamplerGetMagnification : Sampler magnification value

rs_sampler_value rsSamplerGetMagnification(rs_sampler s); Added in API level 16

Get the Sampler's magnification value.

See android.renderscript.S.

rsSamplerGetMinification : Sampler minification value

rs_sampler_value rsSamplerGetMinification(rs_sampler s); Added in API level 16

Get the Sampler's minification value.

See android.renderscript.S.

rsSamplerGetWrapS : Sampler wrap S value

rs_sampler_value rsSamplerGetWrapS(rs_sampler s); Added in API level 16

Get the Sampler's wrap S value.

See android.renderscript.S.

rsSamplerGetWrapT : Sampler wrap T value

rs_sampler_value rsSamplerGetWrapT(rs_sampler s); Added in API level 16

Get the sampler's wrap T value.

See android.renderscript.S.