IVoxelFarmReportLambdaHost

The IVoxelFarmReportLambdaHost interface provides access to platform features.

Methods:

double Input(string id, string label, double defaultValue)	Inputs a floating point value. The parameter provided as id should be unique for the set of inputs in this report lambda.
string InputAttributes(string id, string label, int entity, int type)	Inputs one attribute. The parameter provided as id should be unique for the set of inputs in this report lambda. The parameter entity determines which entity will provide the list of possible attributes.
bool InputBool(string id, string label, bool defaultValue);	Inputs a boolean point value. The parameter provided as id should be unique for the set of inputs in this report lambda.
ulong InputDate(string id, string label, ulong defaultValue);	Inputs a date-time. The parameter provided as id should be unique for the set of inputs in this report lambda.
int InputEntity(string id, string label, int type);	Inputs a entity reference. The parameter provided as id should be unique for the set of inputs in this report lambda.
string InputQuery(string id, string label, int entity);	Inputs a query for the entity specified as parameter. The parameter provided as id should be unique for the set of inputs in this report lambda.
string InputRegion(string id, string label);	Inputs a region. The parameter provided as id should be unique for the set of inputs in this report lambda.
string InputString(string id, string label, string defaultValue);	Inputs a string. The parameter provided as id should be unique for the set of inputs in this report lambda.
int LoadVoxels(int entity, int attributes, string customAttributes);	Requests the platform to load voxels for the provided entity. The attributes parameter contains a bitmask that determines which standard attributes will be loaded for the voxels. Currently only VoxelFarm.SpatialLambda.Attribute.Volume is supported. Custom attributes may contain a comma-separated list of attributes that should be loaded.
void SetAttributes(int entity, string attrib);	Sets which attributes should be loaded for the specified entity.
void SetQuery(int entity, string query);	Sets which query should be applied to the specified entity.

int StartComposite();	Starts a new voxel composite. This method is used in combination with AddLayer(), which inserts a new voxel layer into the voxel composite.
int VoxelsComplement(int componentA, int componentB, int attributes, string customAttributes);	Creates a volumetric boolean complement between two specified voxel layers. The attributes parameter contains a bitmask that determines which standard attributes will be loaded for the voxels. Currently only VoxelFarm.SpatialLambda.Attribute.Volume is supported. Custom attributes may contain a comma-separated list of attributes that should be loaded.
int VoxelsIntersection(int componentA, int componentB, int attributes, string customAttributes);	Creates a volumetric boolean intersection between two specified voxel layers. The attributes parameter contains a bitmask that determines which standard attributes will be loaded for the voxels. Currently only VoxelFarm.SpatialLambda.Attribute.Volume is supported. Custom attributes may contain a comma-separated list of attributes that should be loaded.
int VoxelsUnion(int componentA, int componentB, int attributes, string customAttributes);	Creates a volumetric boolean union between two specified voxel layers. The attributes parameter contains a bitmask that determines which standard attributes will be loaded for the voxels. Currently only VoxelFarm.SpatialLambda.Attribute.Volume is supported. Custom attributes may contain a comma-separated list of attributes that should be loaded.
void AddLayer(int layer, int entity);	Adds a layer to a voxel composite. See the StartComposite() method, which is used in tandem with this function.
float[] GetAttributeBuffer(int component, int attribute, int channel);	Returns a buffer that contains attribute values for the specified voxel component. This function will return null if there is no attribute data for that region of space. The attribute parameter provides the index of the desired attribute. This index is based on the attribute's position in the LoadVoxels() call, or in any of the volumetric boolean calls that also take an attribute list. The channel attribute allows to handle cases where a single voxel contains multiple channels of data for increased precision. See the GetChannelCount() function, which should be called prior to GetAttributeBuffer() to ensure all channels in the voxel are read.
int GetAttributeCount(string attributeList);	Returns the number of individual attributes in the provided attribute list string.
string GetAttributeName(string attributeList, int index);	Returns the attribute name given an attribute index and an attribute list string.
ushort GetChannelCount(int component);	Returns the number of overlapping channels in the voxel data.

float[] GetVolumeBuffer(int component, int channel);	Returns a buffer that contains voxel volumes for the specified voxel component. This function will return null if there is no volume data for that region of space. Volume values range from 0 to 1. In order to compute actual volume in world units, multiply this value by the value returned by GetVoxelVolume(). The channel attribute allows to handle cases where a single voxel contains multiple channels of data for increased precision. See the GetChannelCount() function, which should be called prior to GetAttributeBuffer() to ensure all channels in the voxel are read.
double[] GetVoxelCentroids();	Returns an array of 3D points, where each point corresponds to the centroid of a voxel. This array has three times the number of entries as the arrays returned by GetVolumeBuffer or GetAttributeBuffer, as it takes three entries to represent the coordinates of a single voxel.
double GetVoxelSize();	Returns the length of a voxel in the project.
double GetVoxelVolume();	Returns the volume of a voxel in the project.
int StartList(string name)	Starts a list with the provided name. The function returns an integer handle to the list, which is later used in calls to the List() function.
void List(int listId, string value)	Adds an entry to a list. The listId parameter represents the list. The value parameter holds the contents of the item that will be added to the list.
int StartSum(string entity, string property)	Starts a sum for the provided object and property. This function returns an integer handle for the sum. This handle is later used in calls to Sum().
void Sum(int sumId, double value)	Increments the specified sum by the specified value.

Revision #1
Created 17 March 2025 19:35:04 by admin
Updated 17 March 2025 19:35:37 by admin