以下是使用SYCL对点云数据建立哈希表并存储到iGPU缓冲区的示例代码：

```c++

#include <CL/sycl.hpp>

#include <iostream>

// Define the size of the hash table

constexpr int TABLE\_SIZE = 100;

// Define the maximum number of points to be processed

constexpr int MAX\_POINTS = 1000;

// Define the point structure

struct Point {

float x;

float y;

float z;

};

// Define the hash node structure

struct HashNode {

int key;

Point point;

};

// Define the hash table structure

struct HashTable {

HashNode nodes[TABLE\_SIZE];

};

int main() {

// Create the queue

sycl::queue q{sycl::gpu\_selector{}};

// Create the input data

Point points[MAX\_POINTS];

for (int i = 0; i < MAX\_POINTS; i++) {

points[i].x = i;

points[i].y = i + 1;

points[i].z = i + 2;

}

// Allocate the memory for the hash table on the device

HashTable \*d\_table = sycl::malloc\_device<HashTable>(1, q);

// Initialize the hash table on the device

q.submit([&](sycl::handler &cgh) {

cgh.parallel\_for(sycl::range<1>{1}, [=](sycl::id<1>) {

for (int i = 0; i < TABLE\_SIZE; i++) {

d\_table->nodes[i].key = -1;

}

});

});

// Insert the points into the hash table on the device

q.submit([&](sycl::handler &cgh) {

sycl::accessor<HashTable, 1, sycl::access::mode::write, sycl::access::target::global\_buffer> table\_acc{d\_table, q};

cgh.parallel\_for(sycl::range<1>{MAX\_POINTS}, [=](sycl::id<1> idx) {

int key = idx[0] % TABLE\_SIZE;

while (table\_acc[0].nodes[key].key != -1) {

key = (key + 1) % TABLE\_SIZE;

}

table\_acc[0].nodes[key].key = idx[0];

table\_acc[0].nodes[key].point = points[idx[0]];

});

});

// Print the hash table on the device

q.submit([&](sycl::handler &cgh) {

sycl::accessor<HashTable, 1, sycl::access::mode::read, sycl::access::target::global\_buffer> table\_acc{d\_table, q};

cgh.parallel\_for(sycl::range<1>{1}, [=](sycl::id<1>) {

for (int i = 0; i < TABLE\_SIZE; i++) {

if (table\_acc[0].nodes[i].key != -1) {

std::cout << "Key: " << table\_acc[0].nodes[i].key << ", Point: (" << table\_acc[0].nodes[i].point.x << ", " << table\_acc[0].nodes[i].point.y << ", " << table\_acc[0].nodes[i].point.z << ")" << std::endl;

}

}

});

});

// Free the memory for the hash table on the device

sycl::free(d\_table, q);

return 0;

}

```

在这个示例代码中，我们首先创建了一个SYCL队列，然后定义了哈希表的大小和要处理的点的数量。接下来，我们定义了点和哈希节点的结构，以及哈希表的结构。然后，我们在主函数中创建了输入数据，然后使用`malloc\_device()`函数在设备上为哈希表分配内存，并在设备上初始化了哈希表。然后，我们使用`parallel\_for()`函数将点插入到设备上的哈希表中，并使用访问器来读写iGPU缓冲区。最后，我们使用`parallel\_for()`函数在设备上打印哈希表。最后，我们使用`free()`函数释放了设备上的哈希表内存。

请注意，此示例代码仅用于说明如何使用SYCL对点云数据建立哈希表并存储到iGPU缓冲区。实际使用时，可能需要根据特定需求进行修改。