[使用Kinect生成点云数据](http://blog.csdn.net/hcx25909/article/details/8654684)

        参考台湾Heresy大神的程序:

<http://kheresy.wordpress.com/2011/01/25/build_3d_point_cloud_via_openni/>

**[cpp]** [view plain](http://blog.csdn.net/hcx25909/article/details/8654684) [copy](http://blog.csdn.net/hcx25909/article/details/8654684)

1. #include <XnCppWrapper.h>
2. #include <iostream>
3. #include <iomanip>
4. #include <vector>
6. **using** **namespace** xn;
7. **using** **namespace** std;
9. //point cloud data struct
10. **struct** SColorPoint3D
11. {
12. **float**  X;
13. **float**  Y;
14. **float**  Z;
15. **float**  R;
16. **float**  G;
17. **float**  B;
19. SColorPoint3D( XnPoint3D pos, XnRGB24Pixel color )
20. {
21. X = pos.X;
22. Y = pos.Y;
23. Z = pos.Z;
24. R = (**float**)color.nRed / 255;
25. G = (**float**)color.nGreen / 255;
26. B = (**float**)color.nBlue / 255;
27. }
28. };
30. **void** GeneratePointCloud( DepthGenerator& rDepthGen,
31. **const** XnDepthPixel\* pDepth,
32. **const** XnRGB24Pixel\* pImage,
33. vector<SColorPoint3D>& vPointCloud )
34. {
35. // number of point is the number of 2D image pixel
36. DepthMetaData mDepthMD;
37. rDepthGen.GetMetaData( mDepthMD );
38. unsigned **int** uPointNum = mDepthMD.FullXRes() \* mDepthMD.FullYRes();
40. // build the data structure for convert
41. XnPoint3D\* pDepthPointSet = **new** XnPoint3D[ uPointNum ];
42. unsigned **int** i, j, idxShift, idx;
43. **for**( j = 0; j < mDepthMD.FullYRes(); ++j )
44. {
45. idxShift = j \* mDepthMD.FullXRes();
46. **for**( i = 0; i < mDepthMD.FullXRes(); ++i )
47. {
48. idx = idxShift + i;
49. pDepthPointSet[idx].X = i;
50. pDepthPointSet[idx].Y = j;
51. pDepthPointSet[idx].Z = pDepth[idx];
52. }
53. }
55. // un-project points to real world
56. XnPoint3D\* p3DPointSet = **new** XnPoint3D[ uPointNum ];
57. rDepthGen.ConvertProjectiveToRealWorld( uPointNum, pDepthPointSet, p3DPointSet );
58. **delete**[] pDepthPointSet;
60. // build point cloud
61. **for**( i = 0; i < uPointNum; ++ i )
62. {
63. // skip the depth 0 points
64. **if**( p3DPointSet[i].Z == 0 )
65. **continue**;
67. vPointCloud.push\_back( SColorPoint3D( p3DPointSet[i], pImage[i] ) );
68. }
69. **delete**[] p3DPointSet;
70. }
72. **int** main(**void**)
73. {
74. XnStatus eResult = XN\_STATUS\_OK;
75. **int** i = 0;
77. // init
78. Context mContext;
79. eResult = mContext.Init();
81. DepthGenerator mDepthGenerator;
82. eResult = mDepthGenerator.Create(mContext);
83. ImageGenerator mImageGenerator;
84. eResult = mImageGenerator.Create(mContext);
86. // set output mode
87. XnMapOutputMode mapMode;
88. mapMode.nXRes = XN\_VGA\_X\_RES;
89. mapMode.nYRes = XN\_VGA\_Y\_RES;
90. mapMode.nFPS  = 30;
91. eResult = mDepthGenerator.SetMapOutputMode(mapMode);
92. eResult = mImageGenerator.SetMapOutputMode(mapMode);
94. // start generating
95. eResult = mContext.StartGeneratingAll();
97. // read data
98. vector<SColorPoint3D> vPointCloud;
99. **while** ( !xnOSWasKeyboardHit() )
100. {
101. eResult = mContext.WaitNoneUpdateAll();
102. // get the depth map
103. **const** XnDepthPixel\*  pDepthMap = mDepthGenerator.GetDepthMap();
105. // get the image map
106. **const** XnRGB24Pixel\*  pImageMap = mImageGenerator.GetRGB24ImageMap();
108. // generate point cloud
109. vPointCloud.clear();
110. GeneratePointCloud(mDepthGenerator, pDepthMap, pImageMap, vPointCloud );
112. // print point cloud
113. cout.flags(ios::left);    //Left-aligned
114. cout << "Point number: " << vPointCloud.size() << endl;
115. **for**(i=0;i<vPointCloud.size();i++)
116. {
117. cout << setw(10) << i;
118. cout << "X:" << setw(10) << vPointCloud[i].X;
119. cout << "Y:" << setw(10) << vPointCloud[i].Y;
120. cout << "Z:" << setw(10) << vPointCloud[i].Z;
121. cout << "R:" << setw(10) << vPointCloud[i].R;
122. cout << "G:" << setw(10) << vPointCloud[i].G;
123. cout << "B:" << setw(10) << vPointCloud[i].B <<endl;
124. }
125. }
127. //stop
128. mContext.StopGeneratingAll();
129. mContext.Shutdown();
131. **return** 0;
132. }

            更新版：

**[cpp]** [view plain](http://blog.csdn.net/hcx25909/article/details/8654684) [copy](http://blog.csdn.net/hcx25909/article/details/8654684)

1. #include <XnCppWrapper.h>
2. #include <iostream>
3. #include <iomanip>
4. #include <vector>
5. #include <unistd.h>
6. #include <GL/glut.h>
7. #include <math.h>
9. **using** **namespace** xn;
10. **using** **namespace** std;
12. /\* 点云数据格式 \*/
13. **struct** point\_xyz
14. {
15. **float**  X;
16. **float**  Y;
17. **float**  Z;
18. **float**  R;
19. **float**  G;
20. **float**  B;
22. point\_xyz(XnPoint3D pos, XnRGB24Pixel color)
23. {
24. X = pos.X;
25. Y = pos.Y;
26. Z = pos.Z;
27. R = (**float**)color.nRed / 255;
28. G = (**float**)color.nGreen / 255;
29. B = (**float**)color.nBlue / 255;
30. }
31. };
33. /\* 点云类 \*/
34. **class** point\_cloud
35. {
36. **private**:
37. Context            &context;
38. DepthGenerator     &depth\_generator;
39. ImageGenerator     &image\_generator;
40. XnStatus           result\_val;
41. XnMapOutputMode    map\_mode;
43. **public**:
44. vector<point\_xyz>  cloud\_vector;        //存放一帧图像中的所有点云数据
46. /\* 构造函数\析构函数 \*/
47. point\_cloud(Context &contex, DepthGenerator &depthGenerator, ImageGenerator &imageGenerator)
48. : context(contex), depth\_generator(depthGenerator),
49. image\_generator(imageGenerator), result\_val(XN\_STATUS\_OK) {}
50. ~point\_cloud() { stop(); }
52. /\* 输出模式设置 \*/
53. **inline** **const** XnMapOutputMode get\_default\_output\_mode();
54. **void** set\_output\_mode(**const** XnMapOutputMode &outputMode);
56. /\* 点云数据处理\*/
57. **void**        init();
58. **void**        stop();
59. **void**        updata();
60. **inline** **void** clear();
61. **inline** **void** print();
62. **inline** **int**  size();
64. /\* 错误输出 \*/
65. **inline** **void** printError();
66. };
68. **inline** **const** XnMapOutputMode point\_cloud::get\_default\_output\_mode()
69. {
70. XnMapOutputMode outputMode = {XN\_VGA\_X\_RES, XN\_VGA\_Y\_RES, 30};
71. **return** outputMode;
72. }
74. **void** point\_cloud::set\_output\_mode(**const** XnMapOutputMode &outputMode)
75. {
76. map\_mode.nFPS  = outputMode.nFPS;
77. map\_mode.nXRes = outputMode.nXRes;
78. map\_mode.nYRes = outputMode.nYRes;
79. }
81. **void** point\_cloud::init()
82. {
83. result\_val = context.Init();
84. printError();
86. /\* 从文件中获取数据 \*/
87. result\_val = context.OpenFileRecording("tempRec.oni");
88. printError();
89. result\_val = context.FindExistingNode(XN\_NODE\_TYPE\_DEPTH, depth\_generator);
90. printError();
91. result\_val = context.FindExistingNode(XN\_NODE\_TYPE\_IMAGE, image\_generator);
92. printError();
94. /\* 从设备中获取数据 \*/
95. /\*result\_val = depth\_generator.Create(context);
96. printError();
97. result\_val = image\_generator.Create(context);
98. printError();
99. // set output mode
100. result\_val = depth\_generator.SetMapOutputMode(map\_mode);
101. printError();
102. result\_val = image\_generator.SetMapOutputMode(map\_mode);
103. printError(); \*/
105. /\* 开始生成数据 \*/
106. result\_val = context.StartGeneratingAll();
107. printError();
108. }
110. **void** point\_cloud::stop()
111. {
112. context.StopGeneratingAll();
113. context.Shutdown();
114. }
116. **inline** **void** point\_cloud::print()
117. {
118. **int** i;
119. cout.flags(ios::left);    //Left-aligned
120. cout << "Point number: " << size() << endl;
121. **for**(i=0;i< size();i++)
122. {
123. cout << "X:" << setw(10) << cloud\_vector[i].X;
124. cout << "Y:" << setw(10) << cloud\_vector[i].Y;
125. cout << "Z:" << setw(10) << cloud\_vector[i].Z;
126. cout << "R:" << setw(10) << cloud\_vector[i].R;
127. cout << "G:" << setw(10) << cloud\_vector[i].G;
128. cout << "B:" << setw(10) << cloud\_vector[i].B <<endl;
129. }
130. }
132. **inline** **int** point\_cloud::size()
133. {
134. **return** cloud\_vector.size();
135. }
137. **inline** **void** point\_cloud::clear()
138. {
139. cloud\_vector.clear();
140. }
142. **inline** **void** point\_cloud::printError()
143. {
144. **if** (result\_val != XN\_STATUS\_OK)
145. {
146. printf("Error: %s", xnGetStatusString(result\_val));
147. exit(-1);
148. }
149. }
151. **void** point\_cloud::updata()
152. {
153. result\_val = context.WaitNoneUpdateAll();
154. /\* 获得深度和图像数据 \*/
155. **const** XnDepthPixel\*  pDepth = depth\_generator.GetDepthMap();
156. **const** XnRGB24Pixel\*  pImage = image\_generator.GetRGB24ImageMap();
158. /\* 清零点云向量 \*/
159. clear();
161. /\* 获得点的数量 \*/
162. DepthMetaData   mDepthMD;
163. depth\_generator.GetMetaData(mDepthMD);
164. unsigned **int** uPointNum = mDepthMD.FullXRes() \* mDepthMD.FullYRes();
166. /\* 获得原始的点云数据 \*/
167. XnPoint3D\* pDepthPointSet = **new** XnPoint3D[uPointNum];
168. unsigned **int** i, j, idxShift, idx;
169. **for**( j = 0; j < mDepthMD.FullYRes(); ++j )
170. {
171. idxShift = j \* mDepthMD.FullXRes();
172. **for**( i = 0; i < mDepthMD.FullXRes(); ++i )
173. {
174. idx = idxShift + i;
175. pDepthPointSet[idx].X = i;
176. pDepthPointSet[idx].Y = j;
177. pDepthPointSet[idx].Z = pDepth[idx];
178. }
179. }
181. /\* 将原始数据转换成真实的3D数据 \*/
182. XnPoint3D\* p3DPointSet = **new** XnPoint3D[uPointNum];
183. depth\_generator.ConvertProjectiveToRealWorld(uPointNum, pDepthPointSet, p3DPointSet);
184. **delete**[] pDepthPointSet;
186. /\* 输出点云数据 \*/
187. **for**( i = 0; i < uPointNum; ++ i )
188. {
189. **if**( p3DPointSet[i].Z == 0 )    // 跳过深度为0的数据
190. **continue**;
192. cloud\_vector.push\_back( point\_xyz( p3DPointSet[i], pImage[i] ) );
193. }
194. **delete**[] p3DPointSet;
195. }