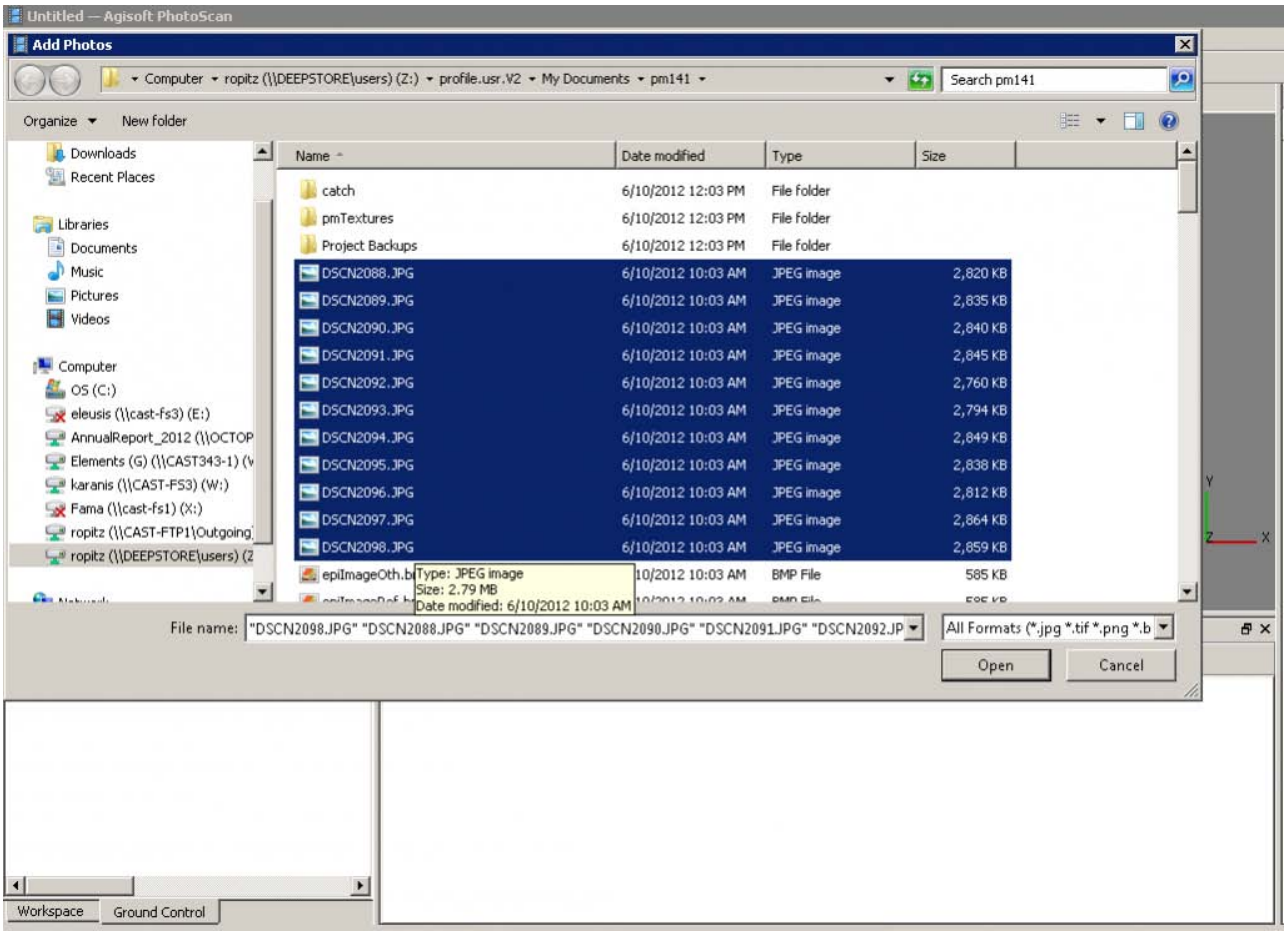
This series will show you how to create 3d models from photographs using Agisoft Photoscan and Esri ArcGIS.
Hint: You can click on any image to see a larger version.

Many archaeological projects now use photogrammetric modeling to record stratigraphic units and other features during the course of excavation. In another post we discussed bringing photogrammetric or laserscanning derived models into a GIS in situations where you don't have precise georeferencing information for the model. In this post we will demonstrate how to use bring a photogrammetric model for which georeferenced coordinates are available, using Agisoft's Photoscan Pro and ArcGIS.

Load Photos

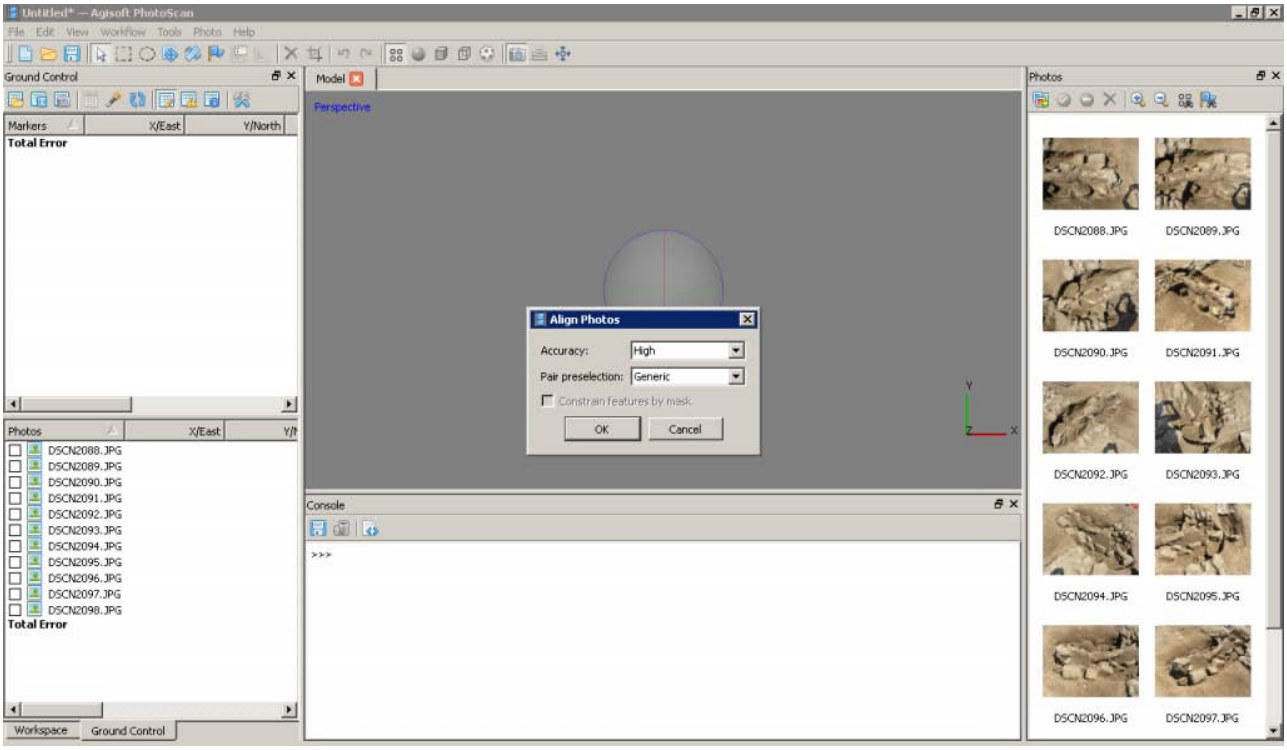
Begin by adding the photos used to create the model to an empty project in Photoscan.



Align Photos

Following the Photoscan Workflow, next align the images. From the menu at the top choose 'Workflow'>'Align Images'. A popup box will appear where you can input the alignment parameters. We recommend selecting 'High' for the accuracy and 'Generic' for the pair pre-selection for most convergent

photogrammetry projects.

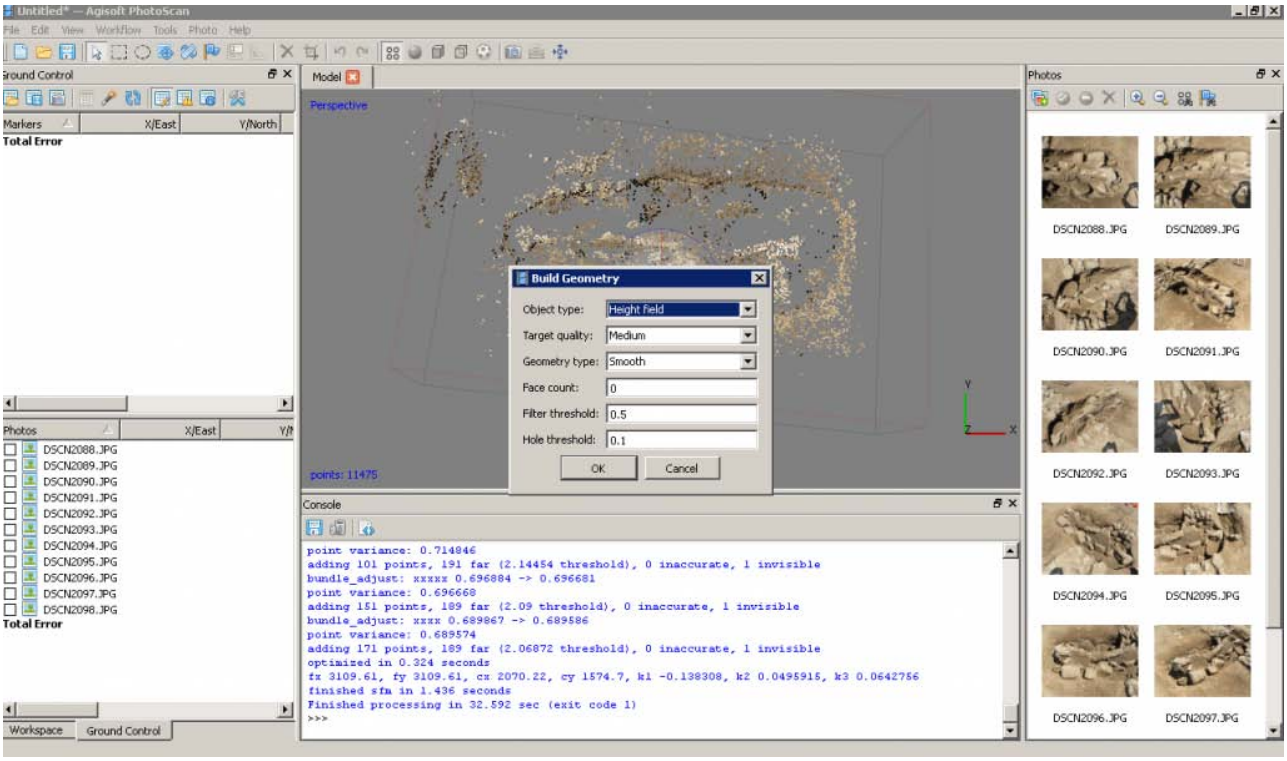


A choice

At this point there are two approaches to adding the georeferenced points to the project. You can place the points directly on each image and then perform the bundle adjustment, or you can build geometry and then place the points on the 3d model, which will automatically place points on each image, after which you can adjust their positions. We normally follow the second approach, especially for projects where there are a large number of photos.

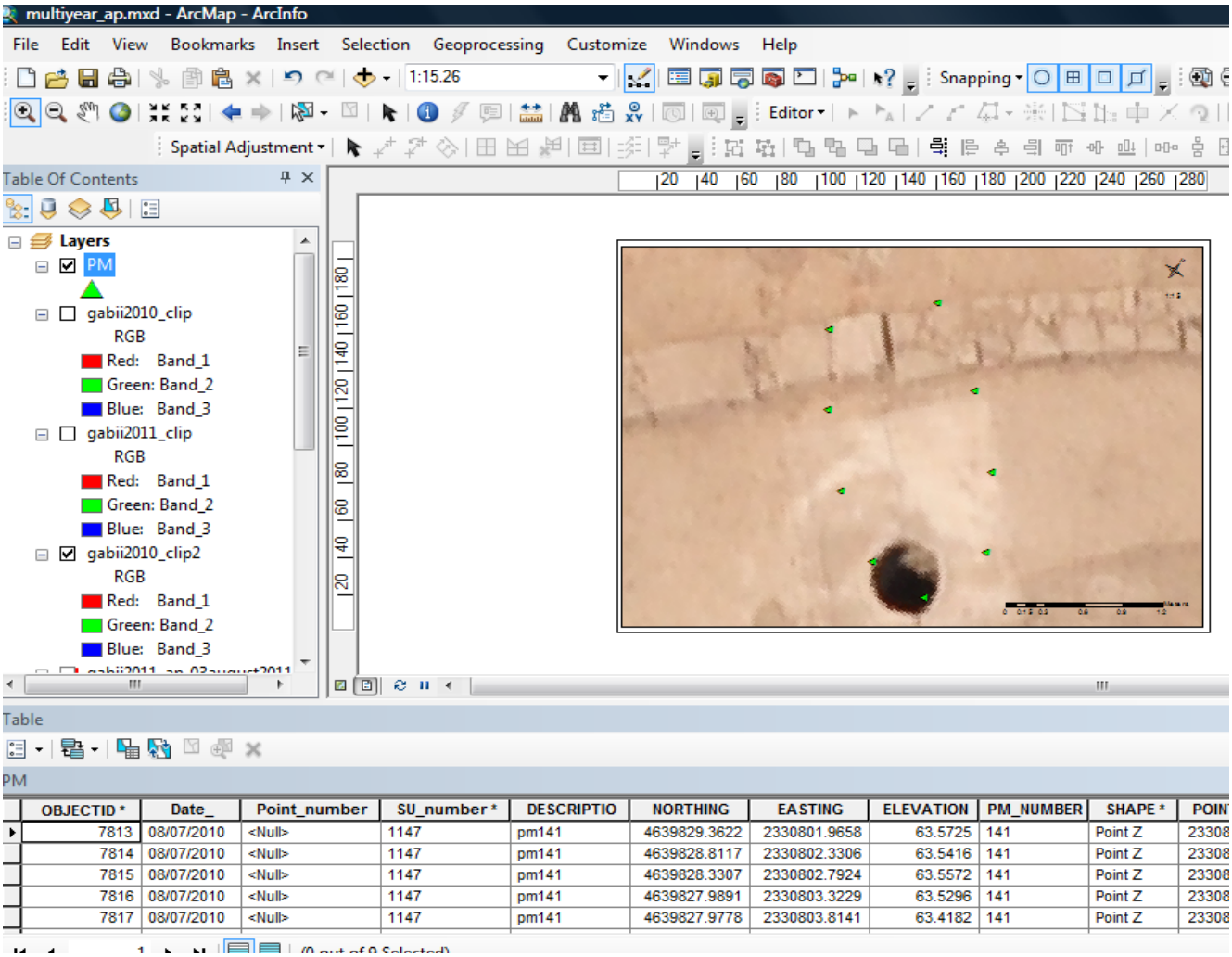
Build Geometry

Under ‘Workflow’ in the main menu, select ‘Build Geometry’. At this point we don’t need to build an uber-high resolution model, because this version of the model is just going to be used to place the markers for the georeferenced points. A higher resolution model can be built later in the process if desired. Therefore either ‘Low’ or ‘Medium’ are good choices for the model resolution, and all other parameters may be left as the defaults. Here we have selected ‘Medium’ as the resolution.



Get the georeferenced points

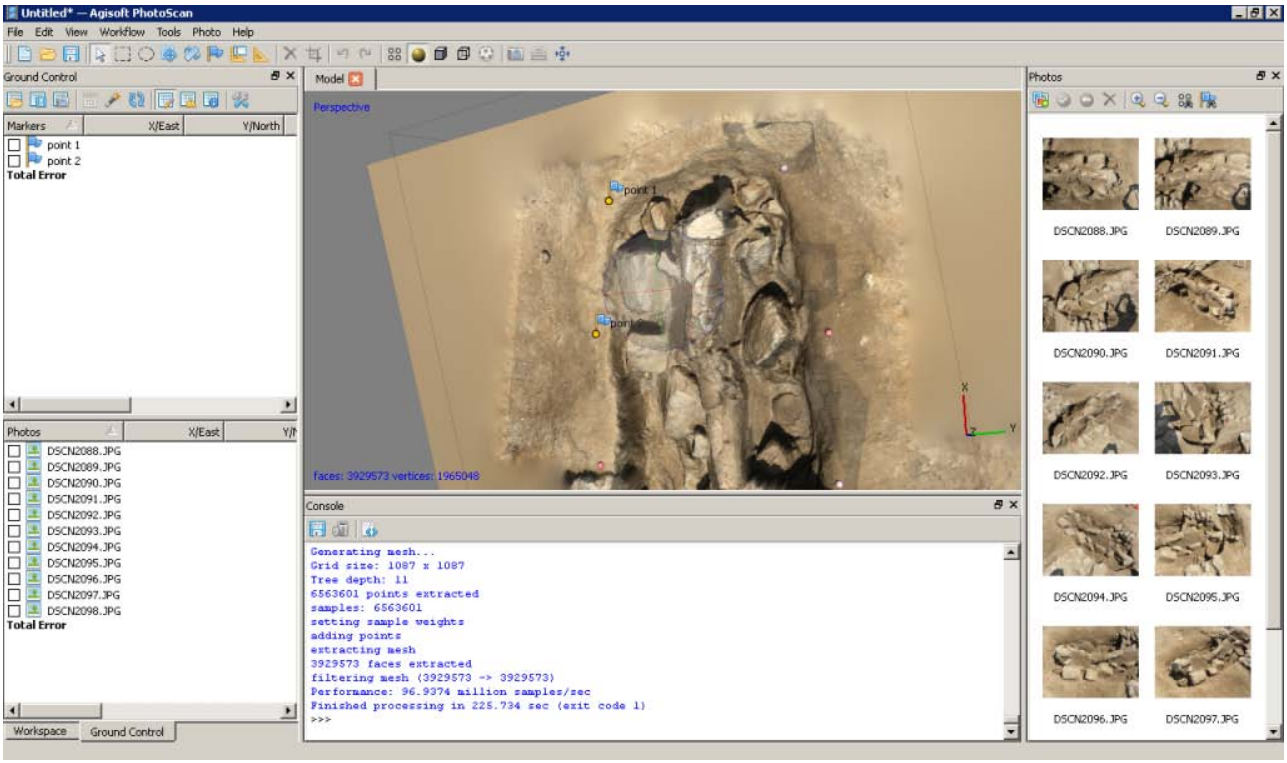
When the photos for this model were taken, targets were places around the feature (highly technical coca-cola bottle caps!) and surveyed using a total station. These surveyed targets are used to georeference the entire model. In this project all surveyed and georeferenced points are stored in an ArcGIS geodatabase. The points for this model are selected using a definition query and exported from ArcGIS.



OBJECTID *	Date_	Point_number	SU_number *	DESCRIPTIO	NORTHING	EASTING	ELEVATION	PM_NUMBER	SHAPE *	POIN
7813	08/07/2010	<Null>	1147	pm141	4639829.3622	2330801.9658	63.5725	141	Point Z	23308
7814	08/07/2010	<Null>	1147	pm141	4639828.8117	2330802.3306	63.5416	141	Point Z	23308
7815	08/07/2010	<Null>	1147	pm141	4639828.3307	2330802.7924	63.5572	141	Point Z	23308
7816	08/07/2010	<Null>	1147	pm141	4639827.9891	2330803.3229	63.5296	141	Point Z	23308
7817	08/07/2010	<Null>	1147	pm141	4639827.9778	2330803.8141	63.4182	141	Point Z	23308

Add the georeferenced points

On the left you have two tabbed menus, ‘Workspace’ and ‘Ground Control’. Switch to the the ‘Ground Control’ menu. Using the ‘Place Markers’ tool from the top menu, place a point on each surveyed target. Enter the corresponding coordinates from the surveyed points through the ‘Ground Control’ menu. *Be careful to check that the northing, easting and height fields map correctly when importing points into Photoscan, as they may be in a different order than in ArcGIS.*

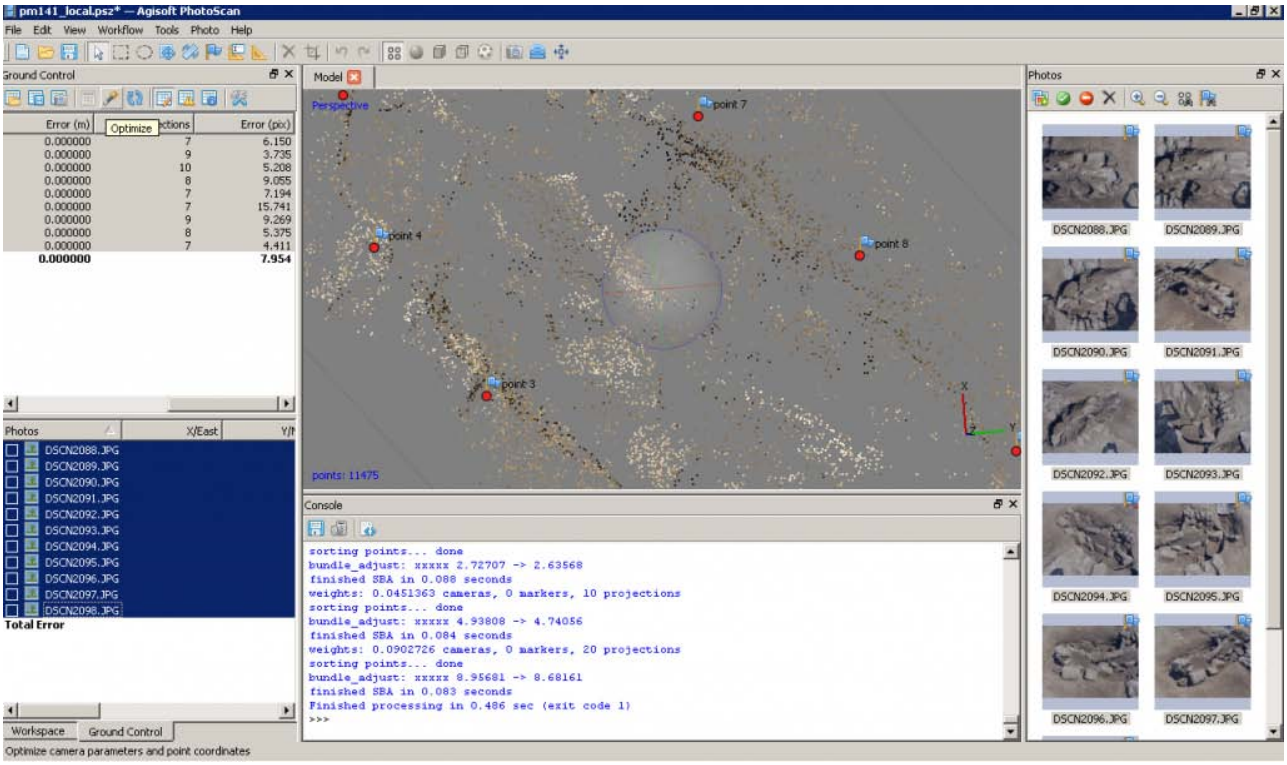


Local coordinates and projections

In practice we have found that many 3d modelling programs don't like it if the model is too far from the world's origin. This means that while Photoscan provides the tools for you to store your model in a real world coordinate system, and this works nicely for producing models as DEMs, you will need to use a local coordinate system if you want to produce models as .obj, .dae, .x3d or other modeling formats and work with them in editing programs like Rapidform or Meshlab. If your surveyed coordinates involve large numbers e.g. UTM coordinates, we suggest creating a local grid by splicing the coordinates so they only have 3-4 pre decimal digits.

Bundle Adjust – Another Choice

After all the points have been placed select all of them (checks on). If you believe the accuracy of the model is at least three time greater than the accuracy of the ground control survey you may select 'update' and the model will be block shifted to the ground control coordinates. If you believe the accuracy of the ground control survey is near to or greater than the accuracy of the model, you should include these points in your bundle adjustment to increase the overall accuracy of the model. To do this select 'optimize' from the 'Ground Control' menu after you have added the points. After the process runs, you can check the errors on each point. They should be less than 20 pixels. If the errors are high, you can attempt to improve the solution by turning off the surveyed points with the highest error, removing poorly referenced photos from the project, or adjusting the location of the surveyed points in individual images. After adjustments are made select 'update' and then 'optimize' again to reprocess the model.



Continue to...

Continue to [PhotoScan – Building Geometry & Texture for Photogrammetry]

]]> http://gmv.cast.uark.edu/photogrammetry/software-photogrammetry/photoscan/photoscan-workflow/photoscan-basic-processing-for-photogrammetry/feed/ 0 http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-setting-up-the-development-environment/ http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-setting-up-the-development-environment/#comments Tue, 11 Dec 2012 11:59:04 +0000 Matt T http://gmv.cast.uark.edu/?p=11061

Using Eclipse IDE

Since there is a plethora of existing tutorials guiding how to set up various development environments in C++, I will show you how to set up the 32-bit OpenNI JAR (OpenNI Java wrapper) in Eclipse IDE and to initialize a production node to begin accessing Kinect data via the Java programming language.
To continue we will be working with the open-source and fantastic piece of software known as Eclipse that you can find here: www.eclipse.org. You will want to download the IDE for Java programmers located on their “downloads” page(about 149 mb). Take note of the many other software solutions that they offer and the vast amount of resources on the site.

> NOTE: Even though we are downloading the “Java” Eclipse IDE you can easily add plugins to use this same piece of software with Python, C/C++, and many other applications.

Additionally, we are assuming that you have already gone through the OpenNI installation located here.

You also need to have the Java JDK installed (www.oracle.com).

Finally, to gain access to one of the best open-source computer vision libraries available, you will need to download and install OpenCV(http://opencv.org/) and the JavaCV(http://code.google.com/p/javacv/). The installation instructions located on each of these sites are excellent.

Setting Up Eclipse with OpenNI: Before You Start

Important Note: As you may already be aware, these tutorials are focused on the Beginner Level user, not only to using the Kinect but also to programming. Before going any further I should also remind you that if jumping "head first" into the new domain of programming isn't something for which you have the interest or the time, there are many things you can accomplish with the "ready to use software" solutions located here.

Also, before starting, make sure that you are using the same platform (32 –bit to 32-bit/64 to 64) on the Eclipse IDE, Java JDK, and OpenNI installation.

Eclipse with OpenNI: Starting a New Java Project

Starting a New Java Project .....

Once you have downloaded Eclipse, installed the java JDK and the OpenNI/Primesense, you will need to start a new Java Project. Following the wizard is the easiest way to do this.

Check the box that says "public static void main(String[] args)" so that Eclipse will add a few lines of code for us.

> **NOTE: For this tutorial I have kept the names fairly vague – be sure to use names that you will remember and understand. Remember that if you use a different naming convention than shown here, you will need to make corrections in the sample code to fit to your specifications.**

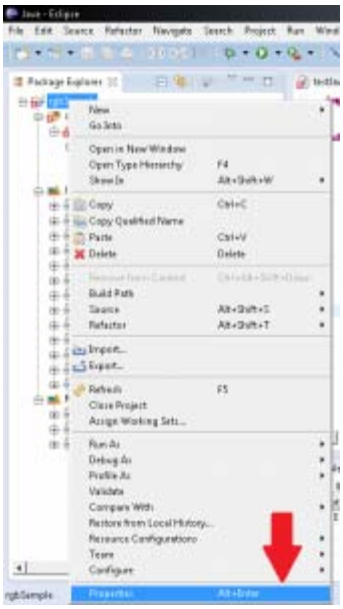
Eclipse with OpenNI: Adding the OpenNI Libraries Part 1

Adding the OpenNI libraries...

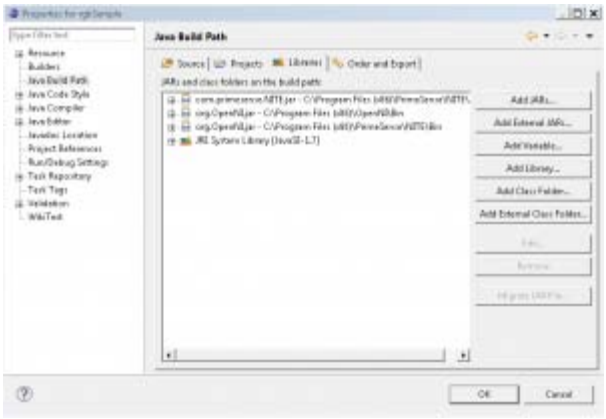
Next we will need to add the OpenNI libraries to the project. This is a pretty straight forward process in Java and Eclipse, simply being a matter of adding the pre-compiled JAR file from the "bin" folder of your OpenNI installation directory.

> NOTE: If you plan on using User Tracking or another Primesense middleware capability you will need to add the JAR in the Primesense directory

To do so right-click on the project we just created:



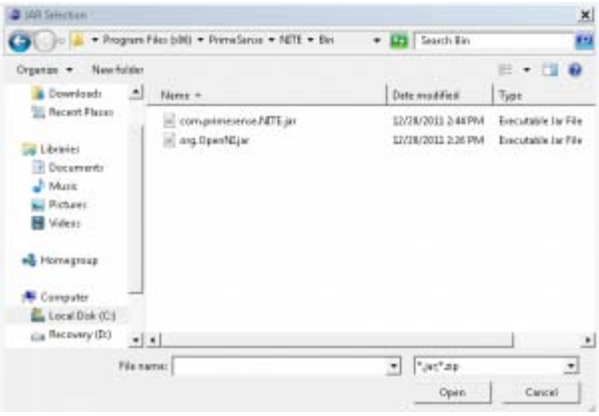
And select the “Properties” menu item.
Then we will want to select the “Java Build Path” and “Add External Jar’s” button.



Repeat the same steps as above for the JavaCV JAR’s that you previously installed somewhere on your machine.

Eclipse with OpenNI: Adding the OpenNI Libraries Part 2

Navigate to the “bin” folders of the install directories for OpenNI and Primesense.
On my Windows 7 64-bit machine with the 32-bit install it is located here:



Note: There are TWO OpenNI "JAR" files – one in the bin folder of the OpenNI install directory as well as one in the Primesense directory. I haven't noticed any difference in using one over the other; as long as your environment paths in Windows are set up to locate the needed files, they should both work.

After this, you should see these files in the "Referenced Libraries" directory on the "Package Explorer" tool bar in Eclipse.

Eclipse with OpenNI: Projects

We should now be able to access the Kinect via Java and Eclipse.

In the following projects we will introduce and attempt to explain the necessary steps for initializing the Kinect via OpenNI and for getting basic access to its data feeds in Java.

Each project goes through setting up the Kinect in OpenNI and includes comments to explain line-by-line what is going on.

For information on Visual Studio 2010 & Microsoft C# SDK....

Using the Microsoft SDK provides a lot of advantages and ease of access, but it is also only applicable to the "Kinect for Windows" hardware and not the Xbox Kinect (as of v1.5).

There are a lot of existing tutorials on the web about setting up your development environment with plenty of sample projects. Below is a list of links to a few of them in no particular order as to avoid reinventing the wheel.

1. http://channel9.msdn.com/Series/KinectQuickstart/Setting-up-your-Development-Environment
2. http://social.msdn.microsoft.com/Forums/el/kinectsdk/thread/7011aca7-defd-445a-bd3c-66837ccc716c
3. http://msdn.microsoft.com/en-us/library/hh855356.aspx
4. Power Point from Stanford

]]> http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-setting-up-the-development-environment/feed/ 0 http://gmv.cast.uark.edu/uncategorized/example-projects-for-the-kinect/ http://gmv.cast.uark.edu/uncategorized/example-projects-for-the-kinect/#comments Thu, 06 Dec 2012 15:32:35 +0000 Matt T http://gmv.cast.uark.edu/?p=11060

Overview

As previously mentioned, the OpenNI API is written in C++ but once you follow the installation procedures covered, here, you will have some pre-compiled wrappers that will give you access to use OpenNI in a few other languages if you need.

Since there is a plethora of existing tutorials regarding setting up various development environments in C++ and corresponding example projects, this article will show you how to setup the 32-bit OpenNI JAR (OpenNI Java wrapper) in Eclipse IDE. We will then initialize an OpenNI production node to begin accessing Kinect data and to get the RGB stream into OpenCv, which is a popular computer vision library.

Before going on to the following project, make sure that you have all of the dependent libraries installed on your machine. For the instructions on getting the 3rd party libraries and for setting up the development environment check out this post.

Also, I want to clarify that this code is merely one solution that I managed to successfully execute. This said, it may have bugs and/or mayb be done more successfully or more easily using a different solution. If you have any suggestions or find errors, please don't hesitate to contact us and I will change the post immediately. These posts follow and continue to follow exploration and collaboration.

Using the Kinect's RGB feed

In this project we will:

1. Make a simple program to capture the RGB feed from the Kinect in Java
2. Get the data into an OpenCV image data structure
3. Display the data on the screen

A high-level overview of the steps we need to take are as follows:

1. Create a new 'context' for the Kinect to be started
2. Create and start a 'generator' which acts as the mechanism for delivering both data and metadata about its corresponding feed
3. Translate the raw Kinect data into a Java data structure to use in native Java libraries
4. Capture a "frame" and display it on screen

The next tab is the commented code for you to use as you wish.

> NOTE: For extremely in-depth and excellent instruction on using JavaCV, the Kinect, along with various other related projects I extremely the book(s) by Andrew Davison from the Imperial College London. A list of his works can be found here http://www.doc.ic.ac.uk/~ajd/publications.html and here http://fivedots.coe.psu.ac.th/~ad/.

Sample RGB Project – Part 1

First, let's import the required libraries:

```
import org.OpenNI.*;

import com.googlecode.javacv.*;
import static com.googlecode.javacv.cpp.opencv_imgproc.*;
import com.googlecode.javacv.cpp.opencv_core.IplImage;
```

Then eclipse nicely fills out our class information.

```
public class sampleRGB {
```

We define some global variables

```
static int imWidth, imHeight;

static ImageGenerator imageGen;
static Context contex
```

Eclipse will also fill out our "main" statement for use by checking the box on the project set up. One addition we will need to make is to surround the following code block with a statement for any exceptions that may be thrown when starting the data feed from the Kinect. Here we are starting a new "context" for the Kinect

```
public static void main(String[] args) throws GeneralException {
```

Create a "context"

```
context = new Context();
```

Sample RGB Project – Part 2

We are manually adding the license information from Primesense. You can also directly reference the xml documents located in the install directory of both the OpenNI and Primesense.

```
License license = new License("PrimeSense", "0KOIk2JeIBYClPWVnMoRKn5cdY4=");
context.addLicense(license);
```

Create a "generator" which is the machine that will pump out RGB data

```
imageGen = ImageGenerator.create(context);
```

We need to define the resolution of the data coming from the image generator. OpenNI calls this mapmode (imageMaps, depthMaps, etc.). We will use the standard resolution.

First initialize it to null.

```
MapOutputMode mapMode = null;
mapMode = new MapOutputMode(640, 480, 30);
imageGen.setMapOutputMode(mapMode);
```

We also need to pick the pixel format to display from the Image Generator. We will use the Red-Green-Blue 8-bit 3 channel or "RGB24"

```
imageGen.setPixelFormat(PixelFormat.RGB24);
```

Sample RGB Project – Part 3

OpenNI also allows us to easily mirror the image so movement in 3d space is reflected in the image plane

```
context.setGlobalMirror(true);
```

Create an Iplimage(opencv image) with the same size and format as the feed from the kinect.

```
IplImage rgbImage = IplImage.create(imWidth,imHeight, 8, 3);
```

Next we will use the easy route and utilize JFrame/Javacv optimized canvas to show the image

```
CanvasFrame canvas = new CanvasFrame("RGB Demo");
```

Now we will create a never-ending loop to update the data and frames being displayed on the screen. Going line by line we will, update the context every time the image generator gets new data.
Set the opencv image data to the byte buffer created from the imageGen.

NOTE: For some reason the channels coming from the Kinect to the opencv image are ordered differently so we will simply use the opencv convert color to set the "BGR" to "RGB". We tell the canvas frame that we created to show image.

Sample RGB Project – Part 4

Finally, we need to also release the Kinect context or we will get an error the next time we try to start a node because the needed files will be locked

```
while (true){
context.waitOneUpdateAll(imageGen);
rgbImage.imageData(imageGen.createDataByteBuffer());
cvCvtColor(rgbImage, rgbImage, CV_BGR2RGB);
canvas.showImage(rgbImage);

canvas.setDefaultCloseOperation(CanvasFrame.EXIT_ON_CLOSE);
}
```

****Note that you can add an argument to the canvas frame to reverse the channels of the image**

]]> http://gmv.cast.uark.edu/uncategorized/example-projects-for-the-kinect/feed/ 0 http://gmv.cast.uark.edu/modeling/software-visualization/unity-software-visualization/workflow-unity-software-visualization/unity-pro-vs-unity-indie/ http://gmv.cast.uark.edu/modeling/software-visualization/unity-software-visualization/workflow-unity-software-visualization/unity-pro-vs-unity-indie/#comments Sat, 11 Aug 2012 11:53:42 +0000 Keenan http://gmv.cast.uark.edu/?p=11463 Continue reading →]]>

Unity Indie is a free version of the Unity3D software. It allows anyone to use a modern advance game engine to create interactive realtime 3D visualizations for Windows, Mac, Web player and soon, Linux.

The chief differences between Unity Indie and Unity Pro lies in the ability to optimize your scenes and the ability to create scable worlds. Its significant to point out that most of these features in Unity Pro can be imitiated to a certain degree in Indie using code and modeling software.

Intro to Static Batching

Graphics cards can process many polygons with relative ease. The texture mapped onto those polygons tend to be the source of rendering and performance problems. Whenever you test out your Unity application by hitting play, Unity renders on screen everthing that is aligned with the camera (or view frustrum). Every object using a different material within the field of vision gets sent to the graphics card for processing. Unity does this in passes. Everything in the back is rendered first working its way to the front.

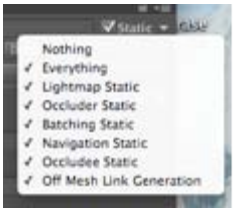
Every object getting sent one by one to the GPU is a fairly inefficient procedure. If there are a number of individual models pointing to the same material, Unity can combine these models at runtime reducing the amount of data that is sent to the GPU. Unity in effect renders or “draws” these combined meshes in one batch or “call”. Unity does this to some degree already with dynamic batching (included in the indie version). But the most significant gains come when you can tell Unity which models you’d like grouped together. This is where static batching comes in (only available in the Pro version).

Tag a Static Batch Object

When you tag an object as static batched, then Unity will group that object in with other objects that share the same material.

Go to Edit > Project Settings > Player

Check Static Batching



Specify the models

Now you’ll want to tell Unity which models should use static batching. Remember, static batching is only effective on models that share the same texture and material. Duplicates of objects and models using texture atlases are your prime targets.

Static Batching Statistics

-Select a prefab in the Project panel or select a GameObject in the Hierarchy.

-In the top right corner of the Inspector, click on the triangle next to Static. A drop down menu appears. Select Batching Static. We’ll talk about some of the other static options in a moment.

-When you click play, you can see how many models are being batched at any given time by click on the Stats button in the upper right corner of the Game panel. You should see a decrease in the Draw Calls also when you enable Static Batching.

Introducing Occlusion Culling

I mentioned above that Unity renders everything in the back first and then everything towards the camera. This means that if you look at a wall in Unity everything behind that wall is getting rendered as well and affecting performance. This seems inefficient and impractical. You could have a sprawling city behind that wall and have it affecting your performance. This is where occlusion culling comes in.

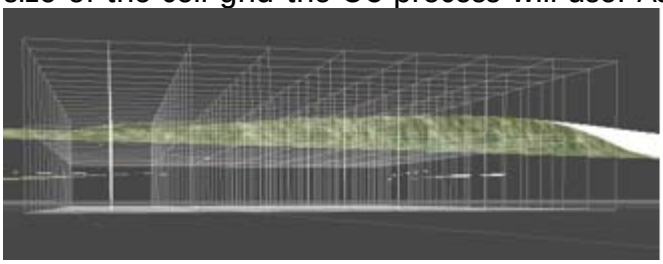
Occlusion culling keeps track of what is visible from any given location via a 3D grid of cells called Potentially Visible Set (PVS). Each cells contains a list of what other cells are visible and which are not. Using this information, Unity can render only that which is visible significantly decreasing the amount of data that needs to be processed to render the scene.

Setting up occlusion culling is fairly straightforward, but keep in mind, depending on the scale of your scene, the baking process could take from 30 minutes to a couple of hours.

Setting up Occlusion Culling

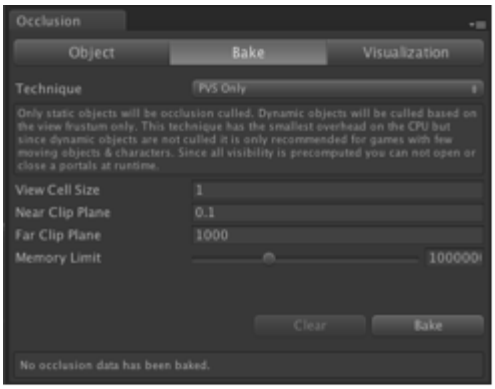
1. Select all the stationary GameObjects
2. In the top right corner of the Inspector, click the triangle next to “Static
3. Select “Occluder Static”
4. Drag and drop your First Person Controller into the scene.
5. Now go to Window > Occlusion Culling

A new window opens called “Occlusion” along with a 3D grid in the scene view. The 3D grid represents the size of the cell grid the OC process will use. As it stands, the entire scene will be used in occlusion culling.



Bake Occlusion Culling

6. Click on the “Bake” tab. You’ll see the settings for the Occlusion Culling.



7. Click on “Bake” and go get some lunch. When the baking finishes, click Play and make sure you do not have geometry that suddenly appears and disappears. You can move the Game view so that it is adjacent to the Scene view. You can see how only the geometry you are viewing in the Game view is the only geometry visible in the scene view and whenever you rotate and move the geometry appears and reappears as needed. This should drastically reduce any frame rate issues.

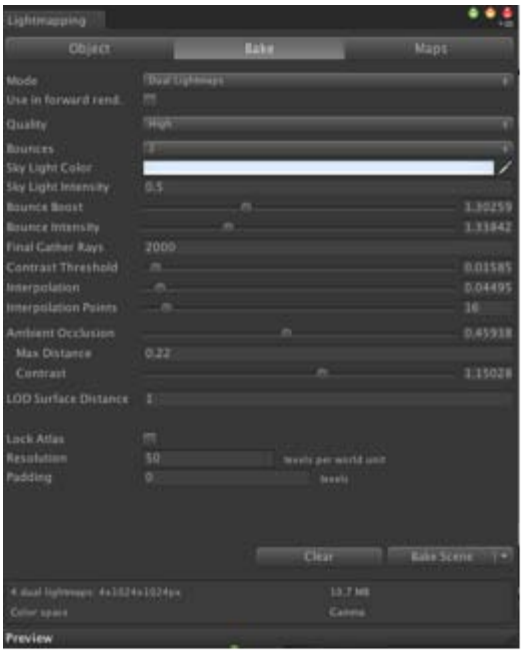
Introducing Lightmapping

When you have lights casting realtime shadows in your scene, the frame rate can be greatly affected as every pixel of an object reflecting light has to be calculated for the right affect and the shadows have to be calculated as well. Lightmapping is a process that “bakes” or draws the shadows and lighting effects onto the textures of an object so that at runtime no light calculations need to be processed. An additional advantage is also that lightmapping can add ambient occlusion to a scene that adds an element of realism and softness to the scene. The setup is similar to Occlusion Culling, but it can take even longer. This is very much an all night process, so be sure to start it before leaving the day, if the scene is large or uses many materials.

As with everything else, you must tag the objects you want to have a lightmap for.

Setting up Lightmapping

1. In the top right corner, click on the triangle next to “Static” and check “Lightmap Static”
2. Go to Window > Lightmapping
3. Click on the Bake tab. The settings for lightmapping appear and at first glance they are daunting. For most situations though, the default settings should do fine. The main thing to look for is to see that “Ambient Occlusion” is at least .4 if you want to add Ambient Occlusion to your scene.
5. Click Bake and call it an evening (depending on the scene size)



]]> http://gmv.cast.uark.edu/modeling/software-visualization/unity-software-visualization/workflow-unity-software-visualization/unity-pro-vs-unity-indie/feed/ 0 http://gmv.cast.uark.edu/modeling/software-visualization/unity-software-visualization/workflow-unity-software-visualization/creating-a-terrain-in-unity-from-a-dem/ http://gmv.cast.uark.edu/modeling/software-visualization/unity-software-visualization/workflow-unity-software-visualization/creating-a-terrain-in-unity-from-a-dem/#comments Sat, 11 Aug 2012 11:31:55 +0000 Keenan http://gmv.cast.uark.edu/?p=11449 Continue reading →]]>

Getting DEMs translated into a form Unity understands can be a bit tricky, and as of yet no perfect solution exists . Nevertheless, DEMs are excellent ways to acquire terrain and elevation models for a variety of purposes. Racing games created in Unity make extensive use of DEMs for levels as well as architectural visualizations. Beware though that accuracy tends to be an issue.

There are two main ways to translate DEM data into Unity. One way is to convert a DEM into a Raw heightmap. The other is to convert a GridFloat dem into a Unity terrain

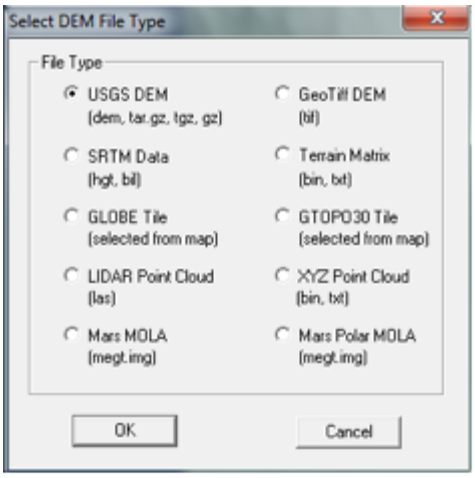
Using Terragen / 3DEM

The goal of this method is to convert the DEM into a RAW heightmap that Unity reads natively. For many users of Unity, the most cost effective solution is to use a two step rocess using Terragen and 3DEM, but if you can obtain a heightmap from DEM by any other means, you can skip to the Import into Unity section.

Convert DEM to Terragen File

The first step is to convert the DEM into a Terragen file. We use the free program 3DEM, which you can download here.

1. Open 3DEM.
2. Choose the format of the DEM in question

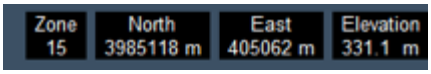


Define the export area

3. Select an area you wish to export.

Extract the export extents

4. Here's the tricky part. The selection box does not give us any information concerning the width, length and elevation of the selected area and Unity will need this information. We will have to point our cursor in severa spots inside the selection box to extract this data. When you move the cursor, the Northing, Easting and Elevation info appears in the lower right hand corner.



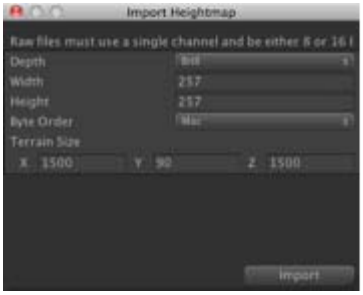
We can use this to extract the width of the selection box. If we point the cursor on the left edge of the box and then the right edge, we subtract the value from each and find the width of the box. Similarly, if we find the lowest elevation and subtract it from the highest elevation, we have our elevation height that Unity will need.
6. File > SaveTerragen Terrain > Selected Area and save it in an accessible area

Export from Terragen to RAW

7. Now open Terragen
8. Open the .ter file you exported from 3DEM. The area you selected should now appear in the Landscape and the Rendering Control dialog.
9. Go to Export > Terrain > Export Make sure it is 8 bit Raw.

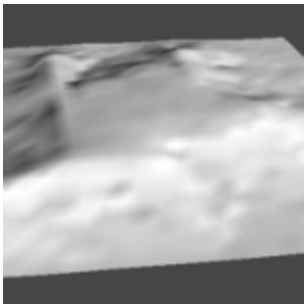
Import into Unity

10. Now open Unity
11. Go to Terrain > Create Terrain. A large terrain will appear in the scene view.
12. Go to Terrain > Import Height – Raw... and select the raw file we exported from Terragen
13. The Import Height dialog appears. You'll need the data we collected from step 4 and enter it into the correct slots. The width and height of the selection box go into the X and Z. The elevation difference goes into Y. Check and make sure the Depth is 8bit, (if you used Terragen) and that the Byte order is your respective OS. Click OK.



The result

Unity will create the terrain from the DEM. In the scene view, the terrain may appear pixelated with jagged edges, but this is mainly due to the Scene view using LOD to render the terrain. If you drop a First Person Controller onto the terrain and hit play, the terrain will appear smooth.



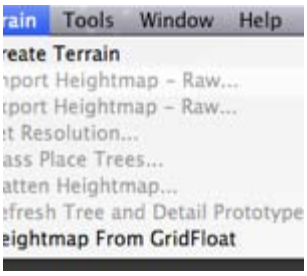
Using GridFloat format

If the DEM is in the GridFloat format, then you can enable Unity to read into a Unity terrain. You'll need to download the script to do this from here. Be sure the header file (HDR) for the GridFloat DEM also resides in the Asset folder.

1. Create a folder in Unity called ‘Editor’
2. Copy the HeightmapFrom GridFloat to the Editor folder.
Placing scripts into an “Editor” folder in Unity actually extends the Unity editor. Now a menu will appear underneath Terrain at the top called ‘Heightmap From GridFloat”

Create the terrain in the scene

3. Select the .flt file in the asset folder. (Be sure you have the hdr file as well)
4. Go to Terrain > Heightmap From GridFloat.



The terrain should appear in the scene view.

]]> http://gmv.cast.uark.edu/modeling/software-visualization/unity-software-visualization/workflow-unity-software-visualization/creating-a-terrain-in-unity-from-a-dem/feed/ 0
http://gmv.cast.uark.edu/modeling/converting-a-3d-model-to-openctm-in-meshlab-for-webgl/
http://gmv.cast.uark.edu/modeling/converting-a-3d-model-to-openctm-in-meshlab-for-webgl/#comments
Thu, 02 Aug 2012 04:45:43 +0000 Keenan http://gmv.cast.uark.edu/?p=11015

What is OpenCTM?

OpenCTM is a new open source file format for 3D objects that boasts impressive compression capabilities. Using OpenCTM, a 90 megabyte model compresses to 9 megabytes. This makes OpenCTM ideal for web delivery. Although there are still many kinks to iron out, the following tutorial explains how to create an OpenCTM from a 3D model and place it on the web using WebGL and JavaScript.

Demo files

The web viewing script is here as a zip file. You will need a functional web server to place the files in. The script loads the model through an XMLHttpRequest, which requires the web page be loaded from an http server. The model won’t load if the web page is opened from the hard drive.

OpenCTM and Meshlab

The easiest way to convert a model to OpenCTM is to use the open source 3D modelling program MeshLab. Once you have imported the mode into MeshLab, we will perform a number of steps to prepare the model for web delivery.

Texture to Vertex Color

1.The current JavaScript doesn’t support textures, but uses vertex color instead. To convert textures to vertex color, in MeshLab go to Filter > Texture to Vertex Color (between 2 meshes)
2.Make sure the Source Mesh and Target Mesh are the same. Then press Apply.
3.Toggle the textures off to check if the conversion was successful. Render > Render Mode > Texture. You

can also find the icon in the main toolbar.
4.If the texture still appears to be shown on the model after toggling the textures off, you know that the Texture to Vertex color was successful.

meshlab
texture
icon

Cleaning the Mesh

The web viewer can be temperamental about mesh geometry. If there are too many holes or rough edges, the script could cause the browser to crash or hang up indefinitely. The solution is to perform a number of steps that will clean the geometry enough for the web viewer script to be satisfied.

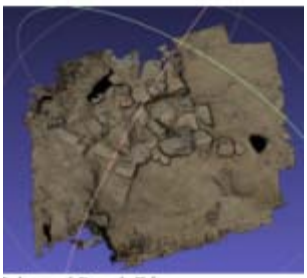
Preliminary Cleaning

1. Filters > Cleaning and Repairing > Close Merged Vertices

2. Filters > Cleaning and Repairing > Remove Duplicated Face

3. Filters > Cleaning and Repairing > Remove Duplicated Vertices

4. Remove Face from Non Manif Edges



oles and Rough Edges

These steps are the preliminary cleaning methods before export. After you complete these steps, export the mesh and try to load it in the browser. If the model loads, all is well. If the browser hangs up or crashes, we'll have to perform additional cleaning steps

Manual Cleaning

For manual cleaning, we need to cut any section of the mesh that appears questionable. Typically this will be near and on the edges of the mesh. The screenshot below gives a good impression of what to look for.

1. Rotate the model so that the section in question is horizontal.

Manual Cleaning 2

2. Click Edit > Select Faces in a Rectangular Region
3. Drag and select the area in question

Manual Cleaning 3

4. Click Filter > Selection > Delete Selected Face and Vertices

5. Repeat for other questionable sections of the mesh
6. Export

Exporting

Whenever you want to give the model a test run, you'll export it out as a OpenCtm file.

1. Go to File > Export Mesh as
2. In the Files of type below, select "OpenCTM compressed file (*.ctm)" from the dropdown men
If OpenCTM is not an available option for export, you may need a more recent version of Meshlab
3. Save as...

Be sure you save the model on the web server where the javascript file is located.

View Your Results

To see the model in the web viewer, we'll need to open up the demo.html file and change one line of code.

1. Open the demo.html file in your favorite editor.

2. On line 72, you will find in quotes the string “changeME.ctm”

3. Change this to the name of the model. If you named the model “myCoolModel.ctm”, you want to add “myCoolModel.ctm” with quotes. If you placed the model in a folder, be sure to add the directory as well, like this “/myModels/myCoolModel.ctm.”

4. Open the website. If you placed it on a local webserver, the address will be something like “http://localhost/demo.html.” if you placed the web scripts and model on a different server, use the name of the server.

NB: Once you have exported as OpenCTM, you can also export out as OBJ for a Unity model as well.

]]> http://gmv.cast.uark.edu/modeling/converting-a-3d-model-to-openctm-in-meshlab-for-webgl/feed/ 0 http://gmv.cast.uark.edu/uncategorized/working-with-data-from-the-kinect/ http://gmv.cast.uark.edu/uncategorized/working-with-data-from-the-kinect/#comments Fri, 06 Jul 2012 19:22:38 +0000 Matt T http://gmv.cast.uark.edu/?p=10459

RGB Image

The color image streaming from the RGB camera is much like that from your average webcam. It has a standard resolution of Height:640 Width:480 at a frame rate of 30 frames per second. You can “force” the Kinect to output a higher resolution image (1280×960) but it will significantly reduce it’s frame rate.

Many things can be done with the RGB data alone such as:

- Image or Video Capture
- Optical Flow Tracking
- Capturing data for textures of models
- Facial Recognition
- Motion Tracking
- And many more….

While it seems silly to purchase a Kinect (about $150) just to use it as a webcam – it is possible. In fact there are ways to hook the camera up to Microsoft’s DirectShow to use it with Skype and other webcam-enabled programs. (Check out this project http://www.e2esoft.cn/kinect/)

- For an Example Project Using the Kinect RGB feed ……

Depth Image

The Kinect is suited with two pieces of hardware, which through their combined efforts, give us the “Depth Image”.
It is the Infrared projector with the CMOS IR “camera” that measures the “distance” from the sensor to the corresponding object off of which the Infrared light reflects.

I say “distance” because the depth sensor of the Kinect actually measures the time that the light takes to leave the sensor and to return to the camera. The returning signal to the Kinect can be altered by other factors including:

- **The physical distance** – the return of this light is dependent on it reflecting off of an object within the range of the Kinect (~1.2–3.5m)

The surface – like other similar technology (range cameras, laser scanners, etc.) the surface which the IR beam hits affects the returning signal. Most commonly glossy or highly reflective, screens (TV,computer,etc.), and windows pose issues for receiving accurate readings from the sensor.

The IR Projector

The IR projector does not emit uniform beams of light but instead relies on a "speckle pattern" according to the U.S. Patent (located here: http://www.freepatentsonline.com/7433024.pdf)

You can actually see the IRMap, as it's called, using OpenNI. Here is a picture from Matthew Fisher's website and another excellent resource on the Kinect (http://graphics.stanford.edu/~mdfisher/Kinect.html)

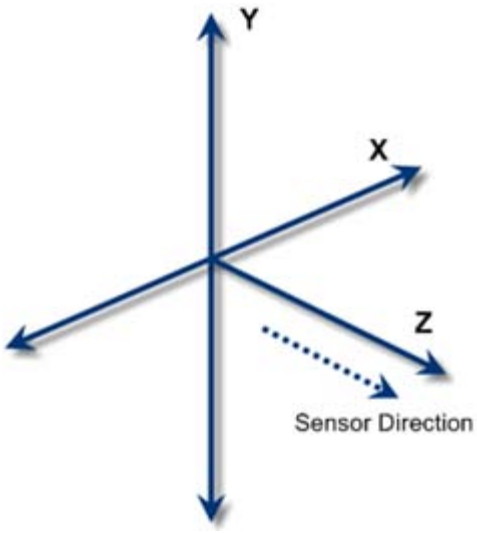
The algorithm used to compute the depth by the Kinect is derived from the difference between the speckle pattern that is observed and a reference pattern at a known depth.

> "The depth computation algorithm is a region-growing stereo method that first finds anchor points using normalized correlation and then grows the solution outward from these anchor points using the assumption that the depth does not change much within a region."

For a deeper discussion on the IR pattern from the Kinect check out this site :
http://www.futurepicture.org/?p=116

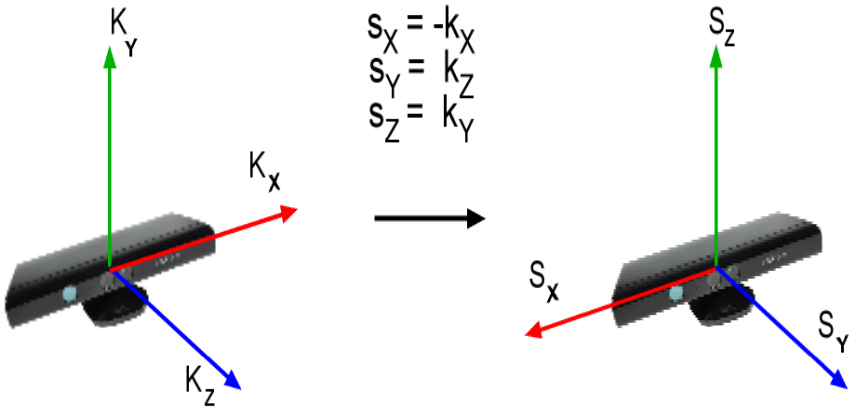
Coordinate System

As one might assume the Kinect uses these "anchor points" as references in it's own internal coordinate system. This origin coordinate system is shown here:



So the (x) is the to left of the sensor, (y) is up, and (z) is going out away from the sensor.

2) Convert from Kinect's coordinates K (k_x, k_y, k_z) to standarized XYZ system S (s_x, s_y, s_z)



This is shown by the (Kx,Ky,Kz) in the above image, with the translated real world coordinates as (Sx,Sy,Sz).

The image is in meters (to millimeter precision).

Translating Depth & Coordinates

So when you work with the Depth Image and plan on using it to track, identify, or measure objects in real world coordinates you will have to translate the pixel coordinate to 3D space.OpenNI makes this easy by using their function (XnStatus xn::DepthGenerator::ConvertProjectiveToRealWorld) which converts a list of points from projective(internal Kinect) coordinates to real world coordinates.

Of course, you can go the other way too, taking real world coordinates to the projective using (XnStatus xn::DepthGenerator::ConvertRealWorldToProjective)

The depth feed from the sensor is 11-bits, therefore, it is capable of capturing a range of 2,048 values. In order to display this image in more 8-bit image structures you will have to convert the range of values into a 255 monochromatic scale. While it is possible to work with what is called the “raw” depth feed in some computer vision libraries(like OpenCV) most of the examples I’ve seen convert the raw depth feed in the same manner. That is to create a histogram from the raw data and to assign the corresponding depth value (from 0 to 2,048) to one of the 255 “bins” which will be the gray-scale value of black to white(0-255) in an 8-bit monochrome image.

You can look at the samples given by OpenNI to get the code, which can be seen in multiple programming languages by looking at their “Viewer” samples.

Accuracy

Another thing worth noting is the difference in accuracy of the depth image as distance from the Kinect increases. There seems to be a decrease in accuracy as one gets further away from the sensor, which makes sense when looking at the previous image of the pattern that the IR projector emits. The greater the physical distance between the object and the IR projector, the less coverage the speckle pattern has on that object in-between anchor points. Or in other words the dots are spaced further apart(x,y) as distance(z) is increased.

The Kinect comes factory calibrated and according to some sources, it isn’t that far off for most applications.

Links for Recalibrating

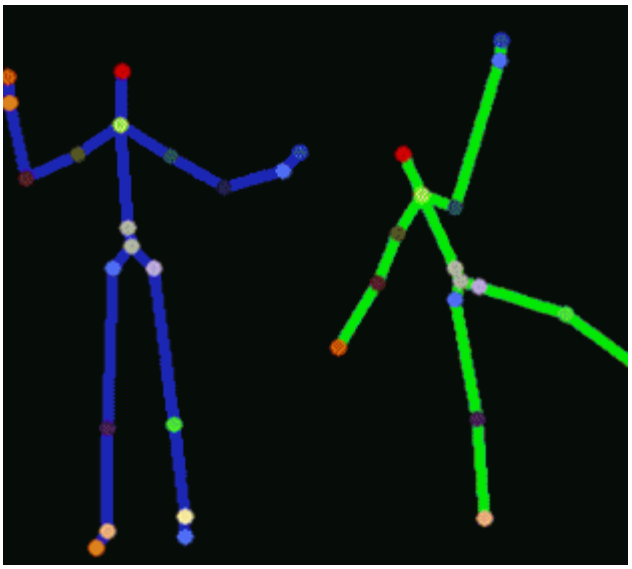
Here are some useful links to recalibrating the Kinect if you want to learn more:

- http://www.ros.org/news/2010/12/technical-information-on-kinect-calibration.html
- http://labs.manctl.com/rgbdemo/index.php/Documentation/TutorialProjectorKinectCalibration
- http://openkinect.org/wiki/Calibration
- http://nicolas.burrus.name/index.php/Research/KinectCalibration
- http://sourceforge.net/projects/kinectcalib/ (a MatLab ToolBox)

User Tracking

The Kinect comes with the capability to track users movements and to identify several joints of each user being tracked. The applications that this kind of readily accessible information can be applied to are plentiful. The basic capabilities of this feature, called “skeletal tracking”, have extended further for pose detection, movement prediction, etc.

According to information supplied to retailers, Kinect is capable of simultaneously tracking up to six people, including two active players, for motion analysis with a feature extraction of 20 joints per player. However,PrimeSense has stated that the number of people that the device can “see” (but not process as players) is only limited by how many will fit into the field-of-view of the camera.



Tracking 2 users **Microsoft

An in depth explanation can be found on the patent application for the Kinect here: http://www.engadget.com/photos/microsofts-kinect-patent-application/

Point Cloud Data

The Kinect doesn't actually capture a 'point cloud'. Rather you can create one by utilizing the depth image that the IR sensor creates. Using the pixel coordinates and (z) values of this image you can transform the stream of data into a 3D "point cloud". Using an RGB image feed and a depth map combined, it is possible to project the colored pixels into three dimensions and to create a textured point cloud.

Instead of using 2D graphics to make a depth or range image, we can apply that same data to actually position the "pixels" of the image plane to 3D space. This allows one to view objects from different angles and lighting conditions. One of the advantages of transforming data into a point cloud structure is that it provides for more robust analysis and for more dynamic use than the same data in the form of a 2D graphic.

Connecting this to the geospatial world can be analogous to the practice of extruding Digital Elevation Models (DEM's) of surface features to three dimensions in order to better understand visibility relationships, slope, environmental dynamics, and distance relationships. While it is certainly possible to determine these things without creating a point cloud, the added ease of interpreting these various relationships from data in a 3D format is self-evident and inherent. Furthermore, the creation of a point cloud allows for an easy transition to creating 3D models that can be applied to various domains from gaming to planning applications.

So with two captured images like this:

Depth Map of the imaged scene, shown left in greyscale.

We can create a 3D point cloud.

Setting Up Your Development Environment

We will give you a few examples on how to set up your Development Environment using Kinect API's.

These are all going to be demonstrated on a Windows 7 64-bit machine using only the 32-bit versions of the downloads covered here.

At the time of this post the versions we will be using are:
OpenNI: v 1.5.2.23
Microsoft SDK: v 1.5
OpenKinect(libfreenect): Not Being Done at this time... Sorry

To use the following posts you need to have installed the above using these directions.

- For Information on setting up Eclipse and the OpenNI/Primesense Java ....

]]> http://gmv.cast.uark.edu/uncategorized/working-with-data-from-the-kinect/feed/ 0
http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-additional-resources/
http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-additional-resources/#comments Fri, 06 Jul 2012 19:13:53 +0000 Matt T http://gmv.cast.uark.edu/?p=10453

Links:Resources & Learning

Resources and Learning

1. **www.kinecthacks.com**
2. **www.kinect.dashhacks.com**
3. **www.kinecteducation.com**
4. **www.developkinect.com**
5. **www.scratch.saorog.com**
6. **www.microsoft.com/education/ww/partners-in-learning/Pages/index.aspx**
7. **blogs.msdn.com/b/uk_faculty_connection/archive/2012/04/21/kinect-for-windows-curriculum**
8. **dotnet.dzone.com/articles/kinect-sdk-resources**
9. **hackaday.com/2012/03/22/kinect-for-windows-resources**
10. **channel9.msdn.com/coding4fun/kinect**

11. www.pcworld.com/article/217283/top_15_kinect_hacks_so_far.html

Links: OpenNI

OpenNI

• openni.org – *Open Natural Interaction, an industry-led, not-for-profit organization formed to certify and promote the compatibility and interoperability of Natural Interaction (NI) devices, applications and middleware*

• github.com/openni – *Open source framework for natural interaction devices*

• github.com/PrimeSense/Sensor – *Open source driver for the PrimeSensor Development Kit*

Links: Tech

Tech

1. www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066 – *Hardware teardown. Chip info is here. (via adafruit)*

2. kinecthacks.net/kinect-pinout – *Pinout info of the Kinect Sensor*

3. www.primesense.com/?p=535 – *Primesense reference implementation (via adafruit thread)*

4. www.sensorland.com/HowPage090.html – *How sensors work and the bayer filter*

5. www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf – *Suggestions to implement pseudocode near the end*

6. http://www.dwheeler.com/essays/floss-license-slide.html – *Which licenses are compatible with which*

7. http://www.eetimes.com/design/signal-processing-dsp/4211071/Inside-Xbox-360-s-Kinect-controller – *Another Hardware Teardown. Note this article incorrectly states that the PS1080 talks to the Marvell chip.*

8. http://nvie.com/posts/a-successful-git-branching-model/ – *Model for branching within Git*

9. http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/SubmittingPatches – *Linux contribution procedure*

10. http://git.kernel.org/?p=git/git.git;a=blob_plain;f=Documentation/SubmittingPatches;hb=HEAD – *Git project contribution procedure*

Hardware Options

Well the first option is to build your own, here's a how-to:

http://www.hackengineer.com/3dcam/

But since not all of us have the time of skills to do that there are other options, like….

1- ASUS Xtion PRO:
Price: $140
Spec's: http://www.newegg.com/Product/Product.aspx?Item=N82E16826785030

2- Leap Motion:
Price: $70
Spec's: https://live.leapmotion.com/about.html

Of course there may be more and there is talk of Sony recently filling a patent resembling their own "Kinect-like" device.

Xbox Kinect vs. Kinect for Windows

As you may know there are actually two "Kinect" sensors out on the market today…

both under the Microsoft company but one was the original made for the Xbox 360 Game console, while the other is the recently released "Kinect for Windows".

Overview

As far as I can tell the two hardware stacks are identical except for the name plate on the front (XBOX or KINECT FOR WINDOWS) and the Windows version has a shorter power cord with a higher price tag due to licensing issues.

There are constant changes being done to the Kinect for Windows to distance itself from it's Xbox twin, like a firmware update to support "Near Mode" in the Windows SDK….

Microsoft even goes as far as saying the following:

> *'The Kinect for Windows SDK has been designed for the Kinect for Windows hardware and application development is only licensed with use of the Kinect for Windows sensor. We do not recommend using Kinect for Xbox 360 to assist in the development of Kinect for Windows applications. Developers should plan to transition to Kinect for Windows hardware for development purposes and should expect that their users will also be using Kinect for Windows hardware as well.'*

If you are currently using the Kinect for Xbox you will find that the automatic registration functions found with the Microsoft SDK will not recognize your Kinect and therefore kick out an error every time you try to run one of their samples.

As far as I know, you can however, still manually register the Kinect with the Microsoft SDK and utilize the functions already developed in the API AT THIS POINT with the XBOX version of the sensor. I wouldn't be surprised if this changes in the near future however.

Published Resources

Published Resources

1) Abramov, Alexey et al. “Depth-supported Real-time Video Segmentation with the Kinect.” Proceedings of the 2012 IEEE Workshop on the Applications of Computer Vision. Washington, DC, USA: IEEE Computer Society, 2012. 457–464. Web. 6 July 2012. WACV ’12.

2) Bleiweiss, Amit et al. “Enhanced Interactive Gaming by Blending Full-body Tracking and Gesture Animation.” ACM SIGGRAPH ASIA 2010 Sketches. New York, NY, USA: ACM, 2010. 34:1–34:2. Web. 6 July 2012. SA ’10.

3) Borenstein, Greg. Making Things See: 3D Vision with Kinect, Processing, Arduino, and MakerBot. Make, 2012. Print.

4) Boulos, Maged N Kamel et al. INTERNATIONAL JOURNAL OF HEALTH GEOGRAPHICS EDITORIAL Open Access Web GIS in Practice X: a Microsoft Kinect Natural User Interface for Google Earth Navigation. Print.

5) Burba, Nathan et al. “Unobtrusive Measurement of Subtle Nonverbal Behaviors with the Microsoft Kinect.” Proceedings of the 2012 IEEE Virtual Reality. Washington, DC, USA: IEEE Computer Society, 2012. 1–4. Web. 6 July 2012. VR ’12.

6) Center for History and New Media. “Zotero Quick Start Guide.”

7) Clark, Adrian, and Thammathip Piumsomboon. “A Realistic Augmented Reality Racing Game Using a Depth-sensing Camera.” Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry. New York, NY, USA: ACM, 2011. 499–502. Web. 6 July 2012. VRCAI ’11.

8) Cui, Yan, and Didier Stricker. “3D Shape Scanning with a Kinect.” ACM SIGGRAPH 2011 Posters. New York, NY, USA: ACM, 2011. 57:1–57:1. Web. 6 July 2012. SIGGRAPH ’11.

9) Davison, Andrew. Kinect Open Source Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java. 1st ed. McGraw-Hill/TAB Electronics, 2012. Print.

10) Devereux, D. et al. “Using the Microsoft Kinect to Model the Environment of an Anthropomimetic Robot.” Submitted to the Second IASTED International Conference on Robotics (ROBO 2011). Web. 6 July 2012.

11) Dippon, Andreas, and Gudrun Klinker. “KinectTouch: Accuracy Test for a Very Low-cost 2.5D Multitouch Tracking System.” Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces. New York, NY, USA: ACM, 2011. 49–52. Web. 6 July 2012. ITS ’11.

12) Droeschel, David, and Sven Behnke. “3D Body Pose Estimation Using an Adaptive Person Model for Articulated ICP.” Proceedings of the 4th International Conference on Intelligent Robotics and Applications – Volume Part II. Berlin, Heidelberg: Springer-Verlag, 2011. 157–167. Web. 6 July 2012. ICIRA’11.

13) Dutta, Tilak. “Evaluation of the Kinect™ Sensor for 3-D Kinematic Measurement in the Workplace.” Applied Ergonomics 43.4 (2012): 645–649. Web. 6 July 2012.

14) Engelharda, N. et al. “Real-time 3D Visual SLAM with a Hand-held RGB-D Camera.” Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden. Vol. 2011. 2011. Web. 6 July 2012.

15) Francese, Rita, Ignazio Passero, and Genoveffa Tortora. “Wiimote and Kinect: Gestural User Interfaces Add a Natural Third Dimension to HCI.” Proceedings of the International Working Conference on Advanced Visual Interfaces. New York, NY, USA: ACM, 2012. 116–123. Web. 6 July 2012. AVI ’12.

16) Giles, J. “Inside the Race to Hack the Kinect.” The New Scientist 208.2789 (2010): 22–23. Print.

17) Gill, T. et al. “A System for Change Detection and Human Recognition in Voxel Space Using the Microsoft Kinect Sensor.” Proceedings of the 2011 IEEE Applied Imagery Pattern Recognition Workshop. Washington, DC, USA: IEEE Computer Society, 2011. 1–8. Web. 6 July 2012. AIPR ’11.

18) Gomez, Juan Diego et al. “Toward 3D Scene Understanding via Audio-description: Kinect-iPad Fusion for the Visually Impaired.” The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility. New York, NY, USA: ACM, 2011. 293–294. Web. 6 July 2012. ASSETS ’11.

19) Goth, Gregory. “Brave NUI World.” Commun. ACM 54.12 (2011): 14–16. Web. 6 July 2012.

20) Gottfried, Jens-Malte, Janis Fehr, and Christoph S. Garbe. “Computing Range Flow from Multi-modal Kinect Data.” Proceedings of the 7th International Conference on Advances in Visual Computing – Volume Part I. Berlin, Heidelberg: Springer-Verlag, 2011. 758–767. Web. 6 July 2012. ISVC’11.

21) Henry, P. et al. “RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments.” The International Journal of Robotics Research (2012): n. pag. Web. 6 July 2012.

22) Henry, Peter et al. “RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments.” Int. J. Rob. Res. 31.5 (2012): 647–663. Web. 6 July 2012.

23) Hilliges, Otmar et al. “HoloDesk: Direct 3d Interactions with a Situated See-through Display.” Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems. New York, NY, USA: ACM, 2012. 2421–2430. Web. 6 July 2012. CHI ’12.

24) Izadi, Shahram et al. “KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera.” Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology. New York, NY, USA: ACM, 2011. 559–568. Web. 6 July 2012. UIST ’11.

25) Jean, Jared St. Kinect Hacks: Creative Coding Techniques for Motion and Pattern Detection. O’Reilly Media, 2012. Print.

26) Kean, Sean, Jonathan Hall, and Phoenix Perry. Meet the Kinect: An Introduction to Programming Natural User Interfaces. 1st ed. Berkely, CA, USA: Apress, 2011. Print.

27) Kramer, Jeff et al. Hacking the Kinect. 1st ed. Berkely, CA, USA: Apress, 2012. Print.

28) LaViola, Joseph J., and Daniel F. Keefe. “3D Spatial Interaction: Applications for Art, Design, and

Science." ACM SIGGRAPH 2011 Courses. New York, NY, USA: ACM, 2011. 1:1–1:75. Web. 6 July 2012. SIGGRAPH '11.

29) Li, Li, Yanhao Xu, and Andreas König. "Robust Depth Camera Based Eye Localization for Human-machine Interactions." Proceedings of the 15th International Conference on Knowledge-based and Intelligent Information and Engineering Systems – Volume Part I. Berlin, Heidelberg: Springer-Verlag, 2011. 424–435. Web. 6 July 2012. KES'11.

30) Livingston, Mark A. et al. "Performance Measurements for the Microsoft Kinect Skeleton." Proceedings of the 2012 IEEE Virtual Reality. Washington, DC, USA: IEEE Computer Society, 2012. 119–120. Web. 6 July 2012. VR '12.

31) Melgar, Enrique Ramos, and Ciriaco Castro Diez. Arduino and Kinect Projects: Design, Build, Blow Their Minds. 1st ed. Berkely, CA, USA: Apress, 2012. Print.

32) Miles, Helen C. et al. "A Review of Virtual Environments for Training in Ball Sports." Computers & Graphics 36.6 (2012): 714–726. Web. 6 July 2012.

33) Miles, Rob. Start Here! Learn the Kinect API. Microsoft Press, 2012. Print.

34) Mitchell, Grethe, and Andy Clarke. "Capturing and Visualising Playground Games and Performance: a Wii and Kinect Based Motion Capture System." Proceedings of the 2011 International Conference on Electronic Visualisation and the Arts. Swinton, UK, UK: British Computer Society, 2011. 218–225. Web. 6 July 2012. EVA'11.

35) Molyneaux, David. "KinectFusion Rapid 3D Reconstruction and Interaction with Microsoft Kinect." Proceedings of the International Conference on the Foundations of Digital Games. New York, NY, USA: ACM, 2012. 3–3. Web. 6 July 2012. FDG '12.

36) Mutto, Carlo Dal, Pietro Zanuttigh, and Guido M. Cortelazzo. Time-of-Flight Cameras and Microsoft Kinect(TM). Springer Publishing Company, Incorporated, 2012. Print.

37) Panger, Galen. "Kinect in the Kitchen: Testing Depth Camera Interactions in Practical Home Environments." Proceedings of the 2012 ACM Annual Conference Extended Abstracts on Human Factors in Computing Systems Extended Abstracts. New York, NY, USA: ACM, 2012. 1985–1990. Web. 6 July 2012. CHI EA '12.

38) Pheatt, Chuck, and Jeremiah McMullen. "Programming for the Xbox Kinect™ Sensor: Tutorial Presentation." J. Comput. Sci. Coll. 27.5 (2012): 140–141. Print.

39) Raheja, Jagdish L., Ankit Chaudhary, and Kunal Singal. "Tracking of Fingertips and Centers of Palm Using KINECT." Proceedings of the 2011 Third International Conference on Computational Intelligence, Modelling & Simulation. Washington, DC, USA: IEEE Computer Society, 2011. 248–252. Web. 6 July 2012. CIMSIM '11.

40) Riche, Nicolas et al. "3D Saliency for Abnormal Motion Selection: The Role of the Depth Map." Proceedings of the 8th International Conference on Computer Vision Systems. Berlin, Heidelberg: Springer-Verlag, 2011. 143–152. Web. 6 July 2012. ICVS'11.

41) Rogers, Rick. "Kinect with Linux." Linux J. 2011.207 (2011): n. pag. Web. 6 July 2012.

42) Shrewsbury, Brandon T. "Providing Haptic Feedback Using the Kinect." The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility. New York, NY, USA: ACM, 2011. 321–322. Web. 6 July 2012. ASSETS '11.

43) Sidik, Mohd Kufaisal bin Mohd et al. "A Study on Natural Interaction for Human Body Motion Using Depth Image Data." Proceedings of the 2011 Workshop on Digital Media and Digital Content Management. Washington, DC, USA: IEEE Computer Society, 2011. 97–102. Web. 6 July 2012. DMDCM '11.

44) Smisek, J., M. Jancosek, and T. Pajdla. “3D with Kinect.” Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference On. 2011. 1154 –1160.

45) Solaro, John. “The Kinect Digital Out-of-Box Experience.” Computer 44.6 (2011): 97–99. Web. 6 July 2012.

46) Stone, E. E, and M. Skubic. “Evaluation of an Inexpensive Depth Camera for Passive In-home Fall Risk Assessment.” Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th International Conference On. 2011. 71–77. Web. 6 July 2012.

47) Sturm, J. et al. “Towards a Benchmark for RGB-D SLAM Evaluation.” Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf.(RSS), Los Angeles, USA. Vol. 2. 2011. 3. Web. 6 July 2012.

48) Sung, J. et al. “Human Activity Detection from RGBD Images.” AAAI Workshop on Pattern, Activity and Intent Recognition (PAIR). 2011. Web. 6 July 2012.

49) Tang, John C., Carolyn Wei, and Reena Kawal. “Social Telepresence Bakeoff: Skype Group Video Calling, Google+ Hangouts, and Microsoft Avatar Kinect.” Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion. New York, NY, USA: ACM, 2012. 37–40. Web. 6 July 2012. CSCW ’12.

50) “The Kinect Revolution.” The New Scientist 208.2789 (2010): 5. Web. 6 July 2012.

51) Tong, Jing et al. “Scanning 3D Full Human Bodies Using Kinects.” IEEE Transactions on Visualization and Computer Graphics 18.4 (2012): 643–650. Web. 6 July 2012.

52) Villaroman, Norman, Dale Rowe, and Bret Swan. “Teaching Natural User Interaction Using OpenNI and the Microsoft Kinect Sensor.” Proceedings of the 2011 Conference on Information Technology Education. New York, NY, USA: ACM, 2011. 227–232. Web. 6 July 2012. SIGITE ’11.

53) “Virtual Reality from the Keyboard/mouse Couple to Kinect.” Annals of Physical and Rehabilitation Medicine 54, Supplement 1.0 (2011): e239. Web. 6 July 2012.

54) Webb, Jarrett, and James Ashley. Beginning Kinect Programming with the Microsoft Kinect SDK. 1st ed. Apress, 2012. Print.

55) Weise, Thibaut et al. “Kinect-based Facial Animation.” SIGGRAPH Asia 2011 Emerging Technologies. New York, NY, USA: ACM, 2011. 1:1–1:1. Web. 6 July 2012. SA ’11.

56) Wilson, Andrew D. Using a Depth Camera as a Touch Sensor. Print.

57) Xu, Yunfei, and Jongeun Choi. “Spatial Prediction with Mobile Sensor Networks Using Gaussian Processes with Built-in Gaussian Markov Random Fields.” Automatica 0 n. pag. Web. 6 July 2012.

58) Zhang, Zhengyou. “Microsoft Kinect Sensor and Its Effect.” IEEE MultiMedia 19.2 (2012): 4–10. Web. 6 July 2012.

]]> http://gmv.cast.uark.edu/uncategorized/microsoft-kinect-additional-resources/feed/ 0
http://gmv.cast.uark.edu/uncategorized/ready-to-use-software-for-the-kinect/
http://gmv.cast.uark.edu/uncategorized/ready-to-use-software-for-the-kinect/#comments Fri, 06 Jul 2012 17:19:17 +0000 Matt T http://gmv.cast.uark.edu/?p=10433

Getting Started Fast with the Kinect

Getting Started

There are a variety of software that are being or have been developed that provide you with access to the Kinect sensor through interactive GUI's. Some use different API's and libraries (OpenNI,Microsoft SDK, etc.) to do this communication with the Kinect and these may conflict with software that you have already installed. So read the sites before downloading and installing. Make sure that you do clean installs.

In this post we will provide you with a high level introduction to two excellent software suites provided on a free-to-use basis.

Instant Access

These provide instant access to the sensor data and help to get your project started whether it involves scanning, recording, viewing, or streaming from the Kinect.

Follow the links at the end of the pages to see an example workflow with each of the two software packages.

RGBDemo

RGB Demo was initially developed by Nicolas Burrus in the RoboticsLab. He then co-founded the Manctl company that now maintains it, helped by various contributors from the opensource community.

Current features

- Grab kinect images and visualize / replay them
- Support for libfreenect and OpenNI/Nite backends
- Extract skeleton data / hand point position (Nite backend)
- Integration with OpenCV and PCL
- Multiple Kinect support and calibration
- Calibrate the camera to get point clouds in metric space (libfreenect)
- Export to meshlab/blender using .ply files
- Demo of 3D scene reconstruction using a freehand Kinect
- Demo of people detection and localization
- Demo of gesture recognition and skeleton tracking using Nite
- Demo of 3D model estimation of objects lying on a table (based on PCL table top object detector)
- Demo of multiple kinect calibration
- Linux, MacOSX and Windows support

RGBDemo can be found here: www.labs.manctl.com/rgbdemo/index.php

Brekel Kinect

In my opinion the Brekel Kinect software is the best that I have seen for easily interfacing with the Kinect.

It uses the OpenNI framework, which we show on the GMV here, but the developer of this software also provides their own packaged OpenNI installer that comes with all the required dependencies.

Current Features

The Brekel Kinect only offers binaries for the Windows platform. It was developed by Jasper Brekelmans in his free time.

It allows you to capture 3D objects and to export them to disk for use in 3D packages. It also allows you to do skeleton tracking which can be streamed into Autodesk's MotionBuilder in realtime, or exported as BVH files.

The greatest things about the Brekel software are that it requires no programming expertise, it has an easy to use GUI, and it has the ability to export almost all of the Kinect's capabilities in a variety of formats.

The Brekel website offers a bunch of links to other resources, downloads, tutorials and answers to FAQ's. Check it out at:

Main Site: http://www.brekel.com

]]> http://gmv.cast.uark.edu/uncategorized/ready-to-use-software-for-the-kinect/feed/ 0 http://gmv.cast.uark.edu/uncategorized/openni-and-the-kinect/ http://gmv.cast.uark.edu/uncategorized/openni-and-the-kinect/#comments Fri, 06 Jul 2012 16:28:24 +0000 Matt T http://gmv.cast.uark.edu/?p=10411

Overview

The OpenNI Organization is primarily supported by PrimeSense, the company who originally developed the Kinect hardware for Microsoft.

The OpenNI framework is not specifically designed to work with the Kinect hardware – rather it is capable of interfacing with various sensors that all happen to be located in the hardware stack known as the Kinect.

OpenNI is primarily written in C++ but comes with Java and .NET wrappers.

Python wrappers do exist, however we have not used them yet.
Here are some links to those wrappers: https://github.com/jmendeth/PyOpenNI, http://code.google.com/p/onipy/

In the next slide we show you how to install the OpenNI modules.

Downloading for Installation 1

This procedure will be focused on a Windows 7 OS machine, with Visual Studio 2010. Although it is a 64-bit machine, we will be downloading the 32-bit versions of the software from OpenNI in order to maintain compatibility throughout our following workflows.

> NOTE: It is generally good etiquette to use 64-bit software on 64-bit machines. However, in this case it seems to provide less pain in the long run to use the 32-bit version as we ultimately intend to deploy our code in programs of projects that use other 32-bit libraries or software. Whether 32-bit or 64-bit, the installation process is identical – you just need to make sure you install all of the right modules for your platform and that you not mix them up by accident.

Download the required packages from the OpenNI website located here:
http://www.openni.org/Downloads/OpenNIModules.aspx

These are the pre-compiled binaries offered by OpenNI for Mac, Windows, and Ubuntu(32 & 64-bit).

You will need to choose the builds you want (Stable or Unstable)and to keep this selection consistent. For assured performance we will be downloading the Stable builds.

Downloading for Installation 2

One last consideration before downloading is ensuring that you download all of the same editions (Development or Redistributable).

For the most part, either one will work. Since we are primarily going to use the Kinect for in-house projects, we are using the development editions.

So we are downloading the following executables:

- **OpenNI Binaries**–> Stable –>OpenNI Stable Build for Windows x86(32-bit) v1.5.2.23 Development Edition
- **OpenNI Compliant Middleware Binaries** –> Stable–> Prime Sense NITE Stable Build for Windows x86(32-bit)v1.5.2.21 Development Edition
- **OpenNI Compliant Hardware Binaries**–> Stable –>Prime Sensor Module Stable Build for Windows x86 (32-bit) v.5.1.0.41

In addition to these you will have to download these drivers:

https://github.com/avin2/SensorKinect/

***On a side note you can skip downloading the individual modules and download one of the packages which includes all three installers in one executable. However, we have had issues in the past with these installing incorrectly. Also note that even when using the packages, you will still need to download the Kinect driver located at the last link above.

Installation

First we need to make sure that the Kinect is unplugged from the computer and that all previous installations have been removed.

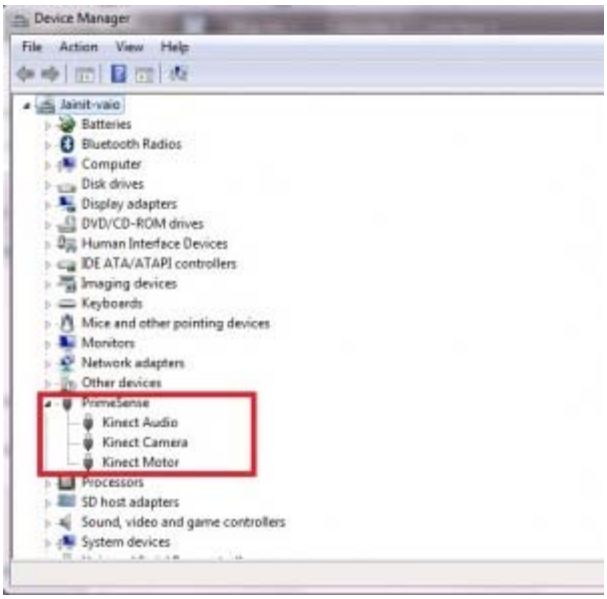
Previously, the order in which you installed these modules was important. For that matter, it may still affect the installation. Regardless, we will continue to follow the convention of installing in this order:

1. “openni-win32XXXX.exe”
2. “nite-win32XXXX.exe”
3. “sensor-win32XXXX.exe”
4. “SensorKinect093-Bin-Win32XXXX.msi” (note this is located in the bin folder of the .zip file we downloaded from https://github.com/avin2/SensorKinect/)

Test the Install

Okay, so now let’s check and confirm that it worked.

Plug in the Kinect and (assuming you are working on a Windows machine) pull up your “Device Manager”. Confirm that you see the following:



If you do not see this or if you have automatic driver installation (meaning that Windows might have installed drivers for the Kinect automatically), make sure that you re-run the installation above and that you completely uninstall the current set of drivers and software.

One final test if the installation is unsuccessful, is to pull up one of the samples provided by OpenNI and Primesense. These are located in the in the installation directory.

For more on using OpenNI check out the other posts on this site and the Additional Resources post.

]]> http://gmv.cast.uark.edu/uncategorized/openni-and-the-kinect/feed/ 0