Kinect Programming for Computer Vision

Outline

- **Organization**
- Introduction to depth cameras and Kinect
- C++ basics
- Assignment

Schedule

- C++ basics, how to use libraries, use OpenNI
- Introduction to OpenGL
- Camera calibration, how depth cameras work
- Object detection
- Human motion analysis
- …
- Project

Requirements to pass the lab course

- Finish the assignments in teams of two until next Wednesday
- Upload a video showing the assignment (use fraps to do the video)
- Finish the final project
- Short exam at the end

Organization

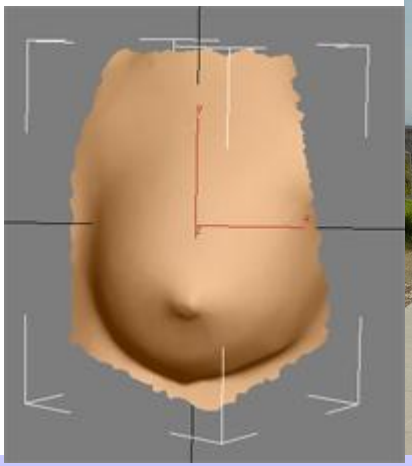
- Teams
- Login
- Assignments

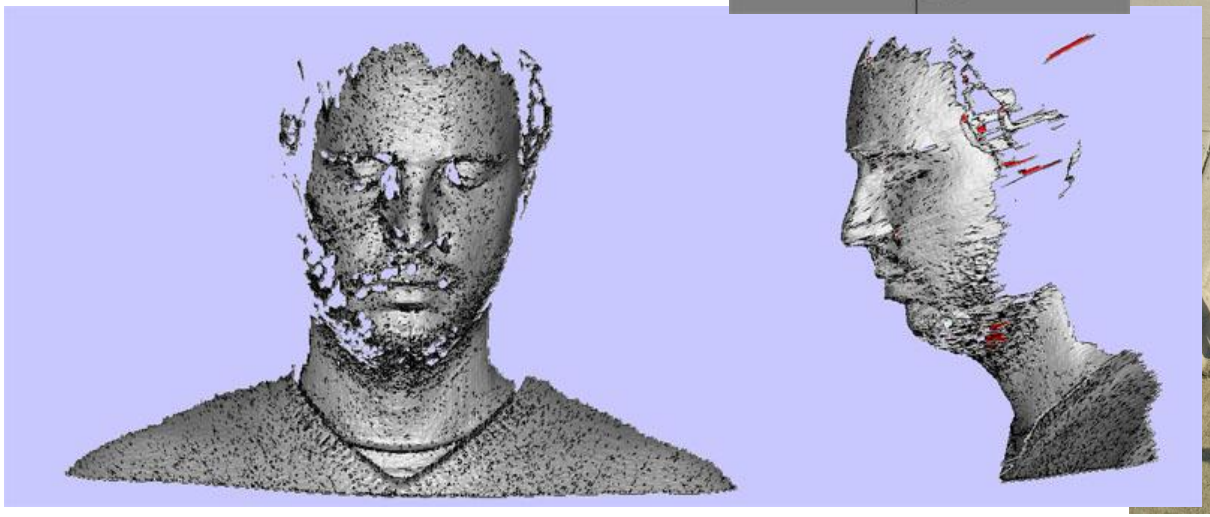
Outline

- Organization
- **Introduction to depth cameras and Kinect**
- C++ basics
- Assignment

Laser rangefinder

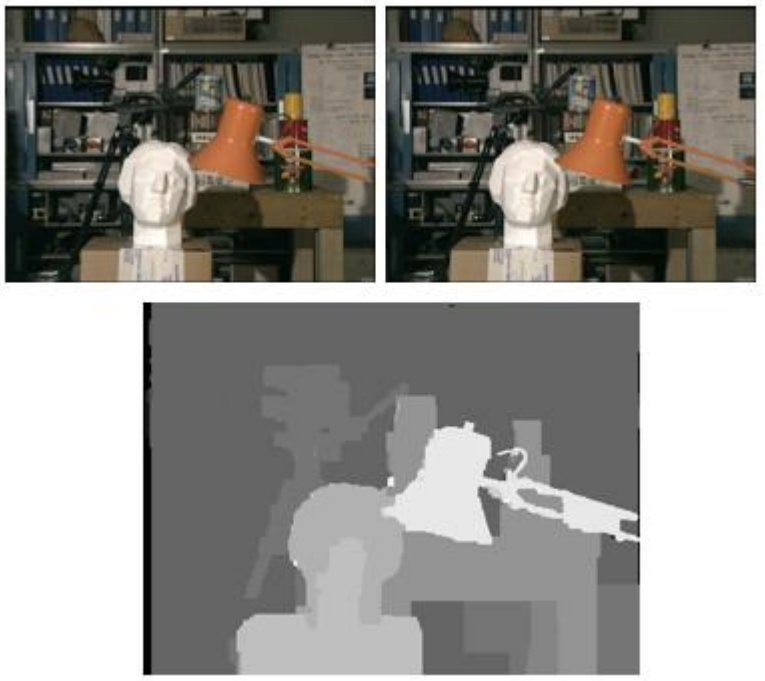
- Time of flight
- Slow
- Construction
- Compare to CAD model
- Plastic Surgery





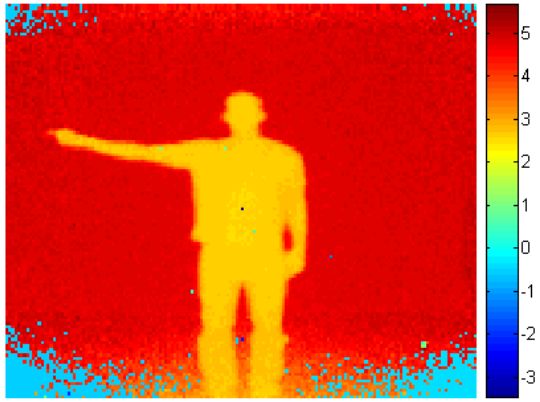
Stereo reconstruction

- Two cameras
- Parallxis



http://www.robots.ox.ac.uk/~pawan/eccv08_tutorial/index.html

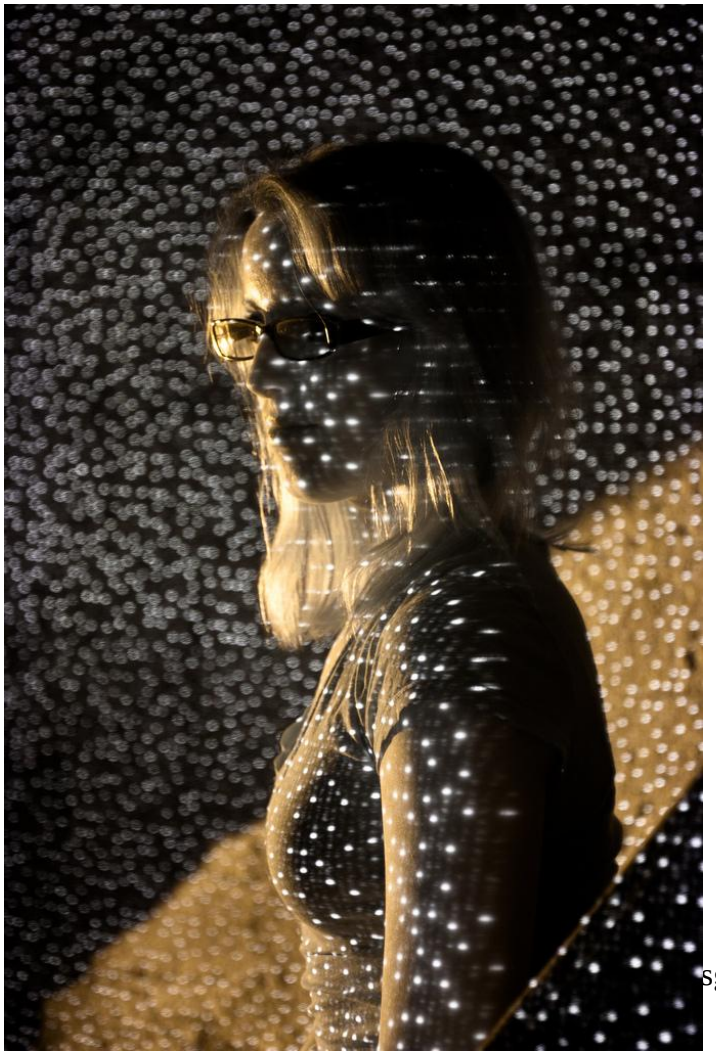
Time of flight



http://www.cvmt.dk/news/colloquia/colloquia2007.html

Structured light

- Project light (visible/invisible) on object
- Stereo or mono
- Kinect
- Audrey Peneven: Dancing With Invisible Light

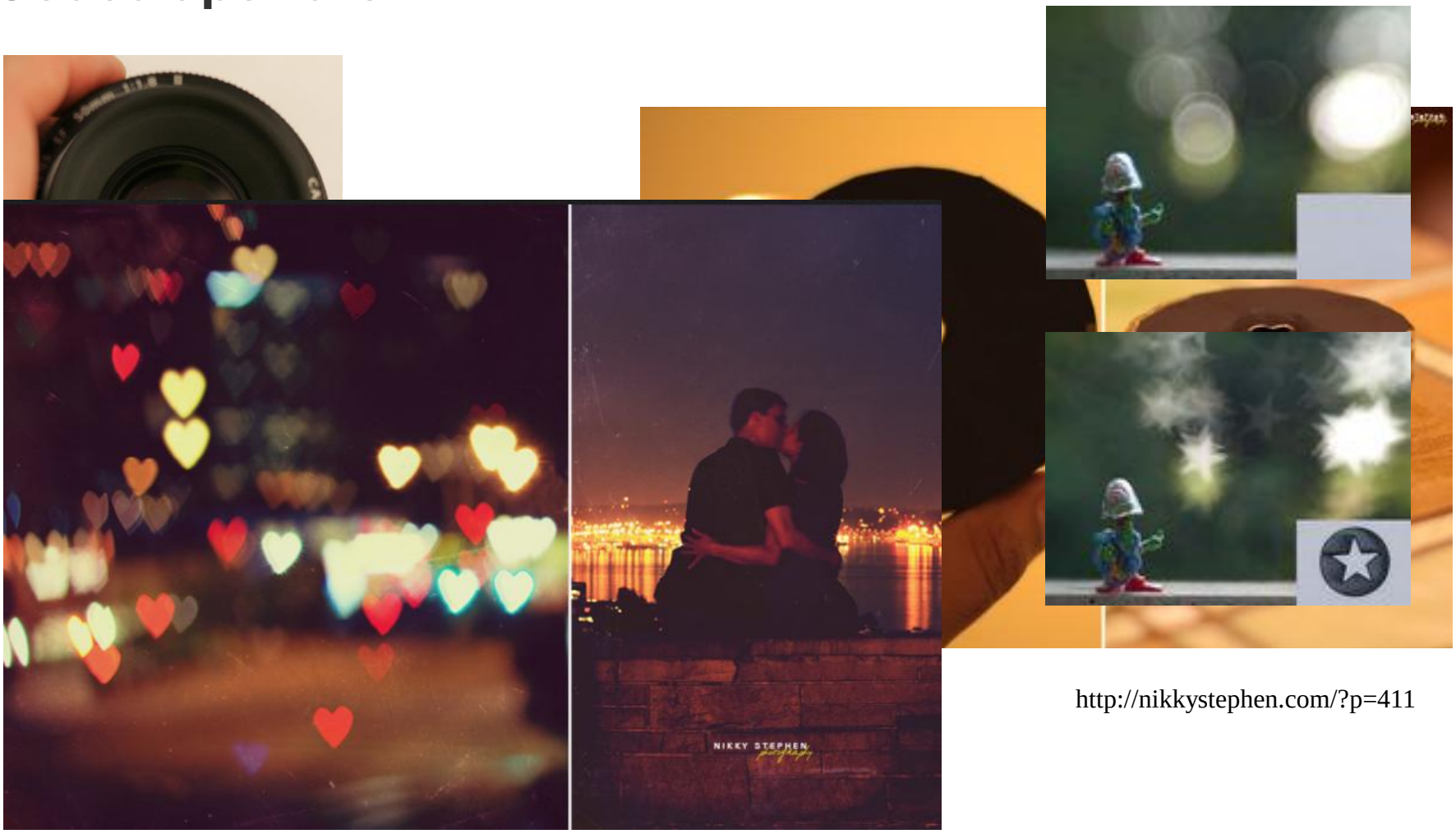


sg

http://audreypenven.net/

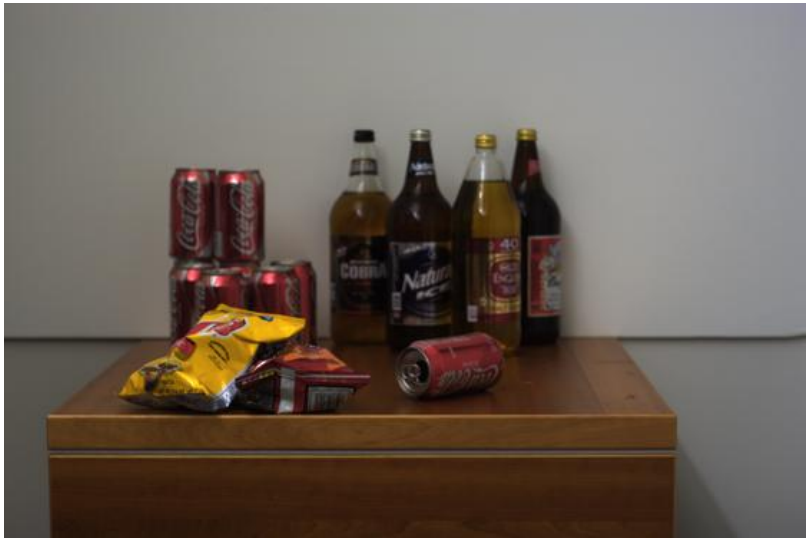
Coded aperture

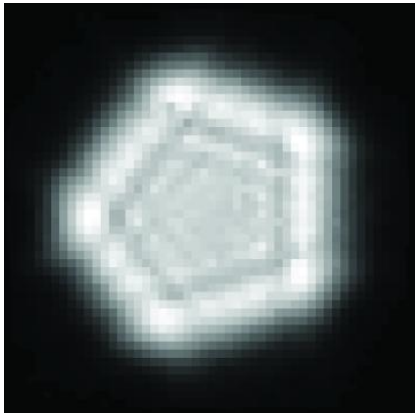
http://olypedia.de/Lensbaby_Creative_Aperture_Kit

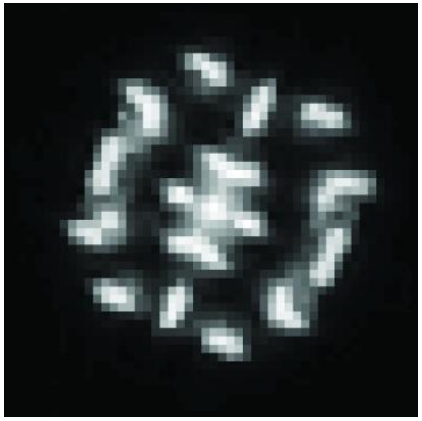


http://nikkystephen.com/?p=411

Coded Aperture

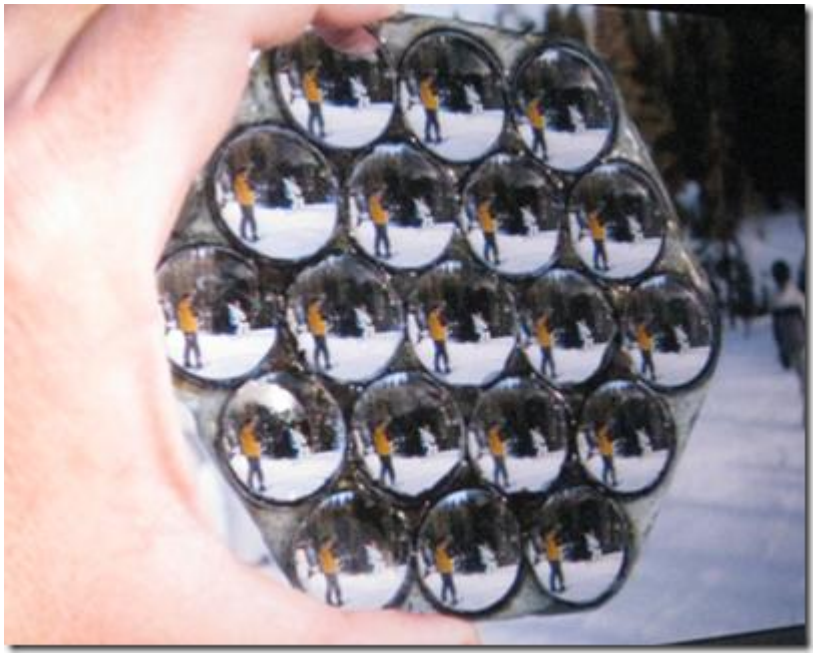






Plenoptic lens

http://lytro.com/gallery/



http://www.itwriting.com/blog/3216-adobes-plenoptic-lens-enables-refocus-magic.html

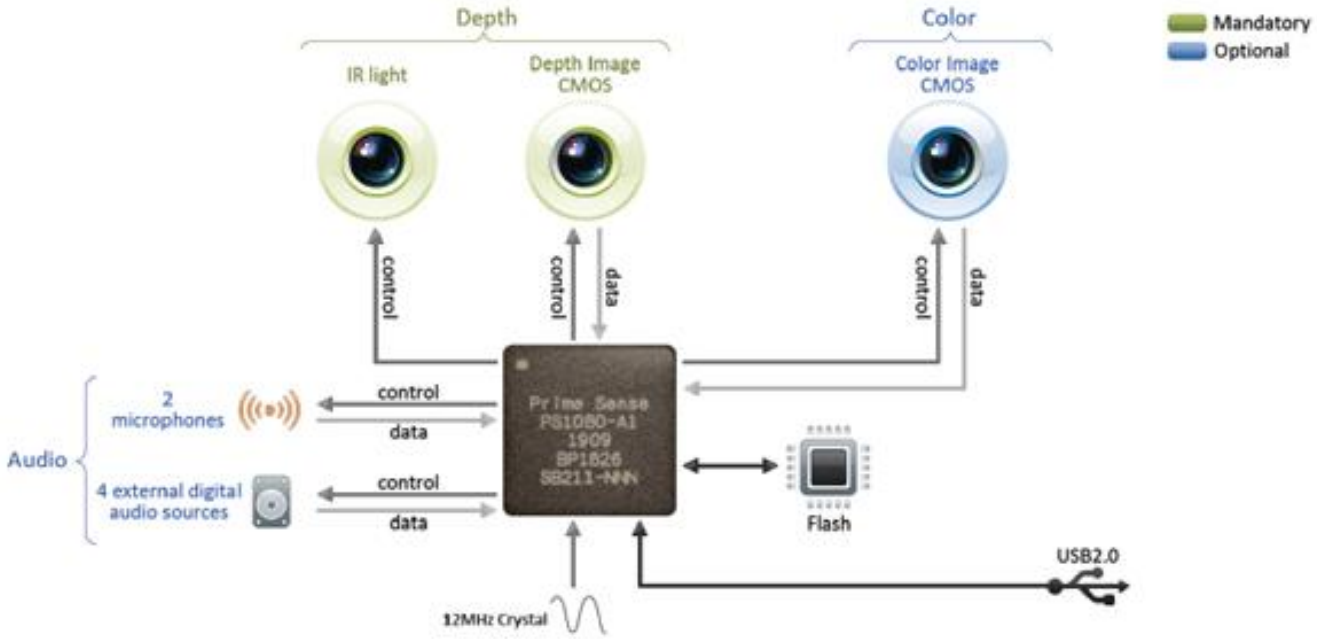
Applications

- Medical
  - Patient position detection
  - Gating
  - Volume registration
- Robotics
- Cars
- Computer Vision
  - Tracking
  - Object detection
  - 3D reconstruction
- User interfaces

http://www.pmdtec.com/markets/medical/

Kinect

- Structured light
- Produced by PrimeSense
- Almost the same as PrimeSensor



Kinect

OpenNI

- Industry-led non-profit organization formed to certify and promote the compatibility and interoperability of Natural Interaction devices, applications and middleware
- Alternative: OpenKinect/libfreenect

PrimeSense™ Natural Interaction

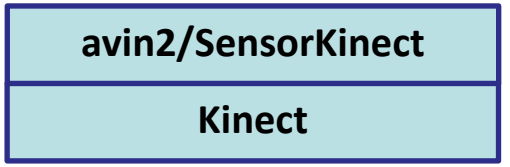
Willow Garage

SIDE•KICK THE LEADER IN MOTION GAMES

ASUS® Inspiring Innovation • Persistent Perfection

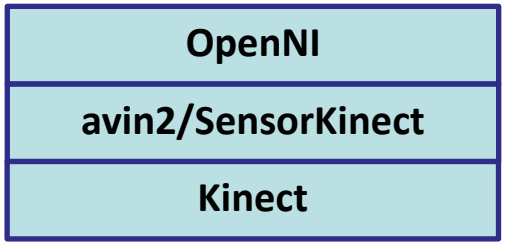
Hardware Drivers

- PrimeSensor drivers available at OpenNI.org
- Do not work with Kinect
- Open Source
- avin2 has modified drivers: https://github.com/avin2/SensorKinect/



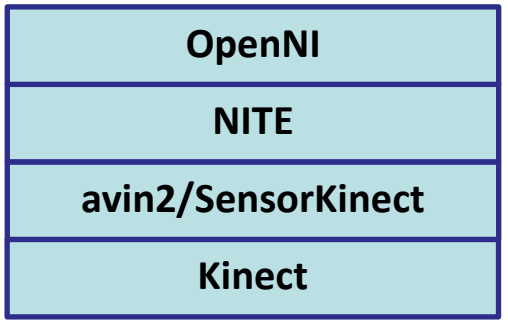
OpenNI framework

- Open Source
- Available at OpenNI.org
- Framework to develop application using depth cameras
- Provides acess to depth image, color image, IR image, auto, gestures, hand tracking, skeleton tracking
- Does not implement everything
- No skeleton tracking



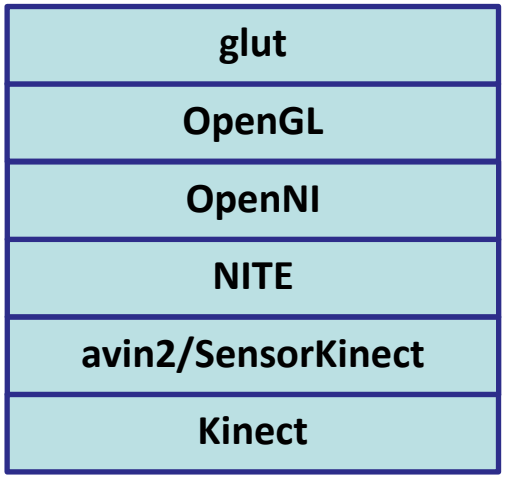
NITE middleware

- Proprietary software by PimeSense for skeleton tracking
- Binaries available at OpenNI.org
- PrimeSense made a license key public
- Is used through OpenNI framework
- Probably not what Xbox 360 uses



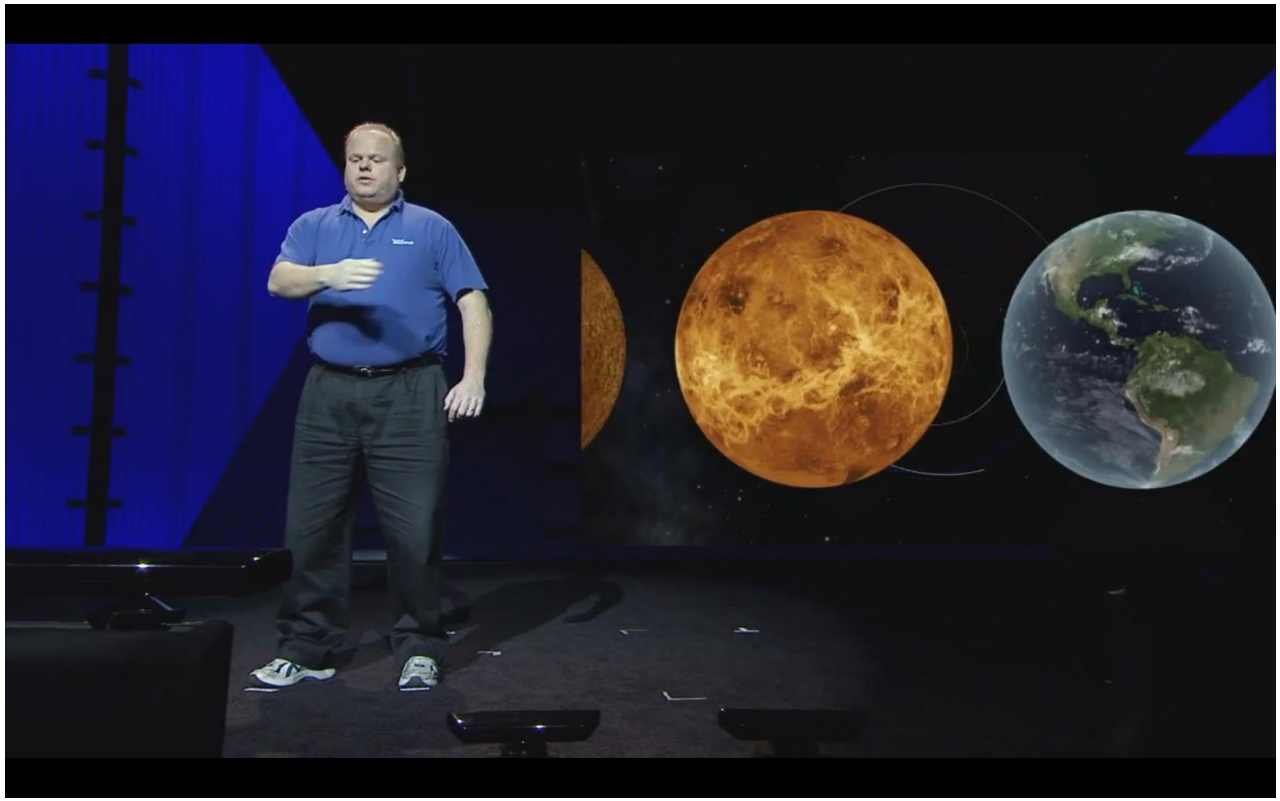
OpenGL / glut / Qt

- 3D graphics: OpenGL / DirectX
- Window manager: glut / Qt / WPF



Kinect for Windows SDK

- Will be released in Spring



Outline

- Organization
- Introduction to depth cameras and Kinect
- **C++ basics**
- Assignment

Simple data types

- **int**: whole number
- **unsigned int**: natural number
- **bool**: boolean value
- **double, float:** floating point values
- **char**: byte, character

```
int phonenumber;
phonenumer = 28917058;

double temperature = 10.3;

bool summertime = true;
```

Arrays and structures

- **Array** : several elements of the same time

```
int phonenumbers[10];
phonenumbers[0] = 28917058;
```

- **Struct** : several elements of arbitrary types

```
struct PhonebookEntry {
        string Lastname;
        string Givenname;
        int    Prefix;
        int    Phonenumber;
};
```

```
PhonebookEntry.Lastname = „Ryu";
PhonebookEntry.Givenname = "Hilla";
PhonebookEntry.Prefix = 089;
PhonebookEntry.Phonenumber = 28917058;
```

- **Enum** : enumeration of specific values (is used like int `int`)

```
enum TrackingType {
        Optical,
        Magnetic,
        Ultrasound
};
```

```
void track(TrackingType tt) {
 if (tt == Optical)
        trackOpt();
 }
```

Header-Files

- All definitions and declarations that are implemented in a .cpp-file

1. Class declaration
2. Typ definition
3. Enumeration
4. Function declarations
5. Constant definitions
6. Member definitions
7. Preprocessor definitions

```
class MyClass { ... };
```

```
struct Position { int x, y };
```

```
enum TrafficLight { red, yellow, green };
```

```
int rectangle(int w, int h);
```

```
const float PI = 3.141593;
```

```
int m_number;
```

```
#include <iostream>
#define VERSION 12
#ifdef __cplusplus
```

Header-files

myClass.h

```
class MyClass : public MyParentClass
{ // the class inherits from MyParentClass
public: // everyone can access this
        MyClass(); // standard constructor
        MyClass(std::string text); // alternative constructor
        virtual ~MyClass(); //destructor

        virtual int func()=0; // virtual (=abstract) function

protected: // only accessible in same or inherited classes
         virtual int fun();

private: // only accessible in this class
         void fu();
         std::string m_someString; // a member variable
}
```

Implementation

myClass.cpp

```
#include “myClass.h”

MyClass::MyClass(){
        m_someNumber = 5;
        m_someString = “initial text”;
}

MyClass::MyClass(std::string text){
        m_someNumber = 5;
        m_someString = text;
}

MyClass::~MyClass(){}

void MyClass::fu(){}
int MyClass::fun(){return m_someNumber;}
double MyClass::funct(){return 2.0;}
```

Pointer

- Memory location of an object

```
Phonebook t;
string searchTerm;
t.addEntry(…);
t.addEntry(…);
…

Entry e;
e = search(searchTerm, t);
```

```
Entry search(string s, Phonebook t)
{ // copies the whole phonebook
    ...
    return e;
}
```

```
Entry search(string* s, Phonebook* t)
{  // copies only 4 byte
    ...
    return e;
}
```

Pointer conversion

- `&` (address operator) :
  “I want to have the memory location of the object”

- `*` (dereference operator):
  „I want to have the the object of a pointer “

```
MyClass object;
MyClass* objectPointer;

objectPointer = &object;
object = *objectPointer;
```

Pointers

- every `new` needs a `delete`

```
MyClass* object = new MyClass();
:
delete object;
```

- One should initialize every pointer with 0 and check on 0 before doint `delete` on it

```
MyClass* object = 0;
if(object)
   delete object;
```

- `new MyClass[42]` needs `delete[]`

Microsoft VS

Debugger

- Use F9 to set break points
- Use F10 to step trough the code

Preprocessor, Linker, Compiler, Libraries

Compiler

1. Preprocessor
   Text replacement based on preprocessor definitions

2. Compiler
   Translated C++ code to binary code

3. Linker
   Links binary program parts together

Preprocessor

- Text replacement *before* compiling
- All preprocessor definitions start with #
  - `#include "file.h"` adds the header-file file.h
  - `#define DEBUGOUT 1` replaces DEBUGOUT with 1 in the program code
  - `#ifdef DEBUGOUT`
    Code1
    `#else`
    Code2
    `#endif`
    inserts Code1 if DEBUGOUT was defined, Code2 if not

class1.cpp

```
#ifdef _WIN32
#pragma warning ( disable : 4244 )
#endif
```

Preprocessor

class1.h

```
int m_myInt;
…
```

class2.h

```
#include "class1.h"
…
```

class3.h

```
#include "class1.h"
#include "class2.h"
…
```

- `m_myInt` is defined two times
- Using preprocessor definitions we can avoid this

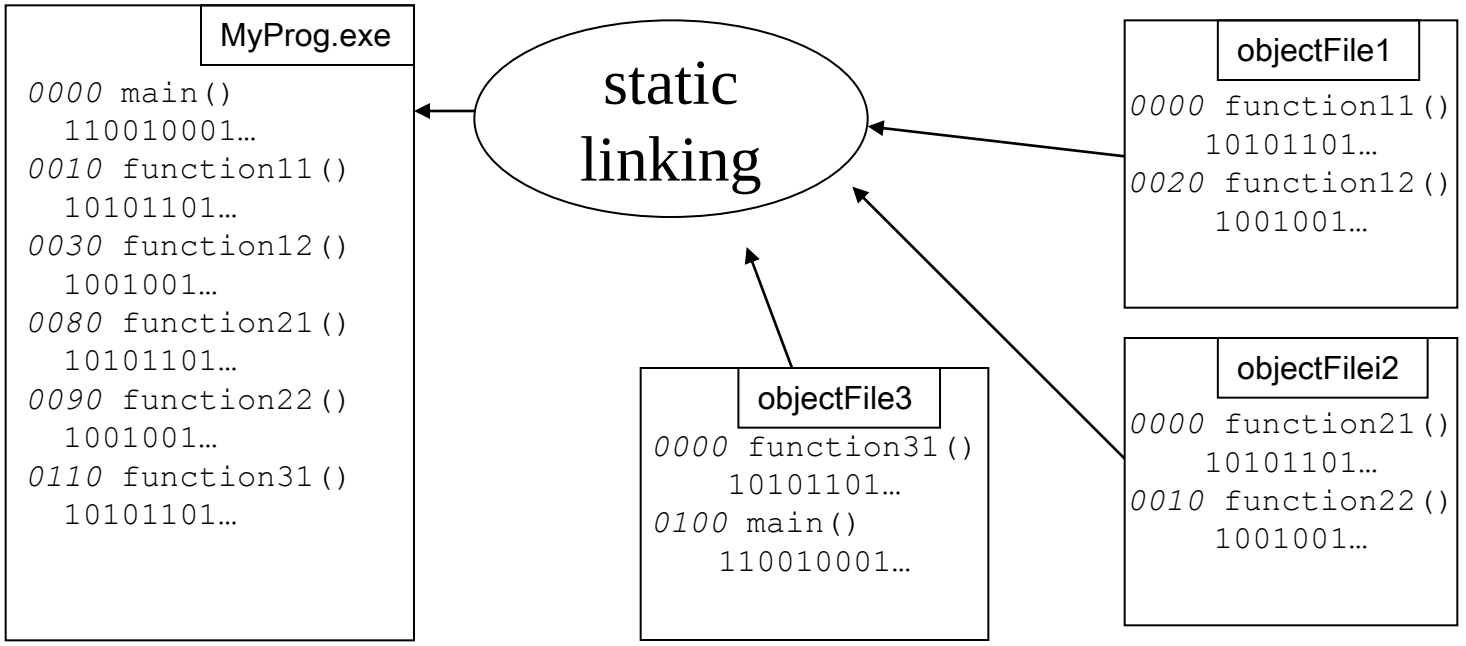
class1.h

```
#ifndef CLASS1_H
#define CLASS1_H
int m_myInt;
…
#endif
```

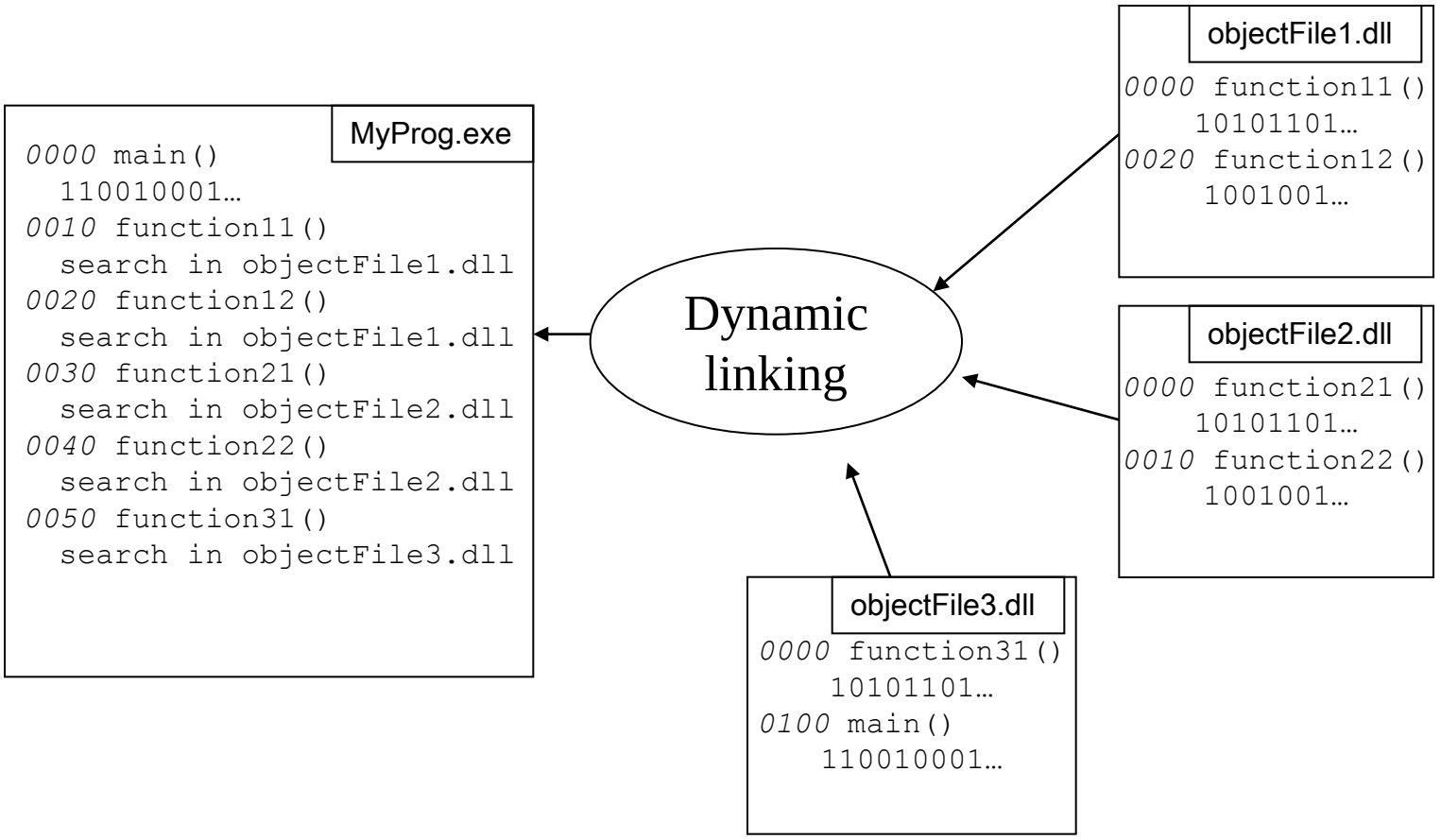
Libraries

- Libraries (*.lib) are compiled binary files without entry point
- Faster, as not all parts of a program have to be recompiled every time
- Proprietary programs can be distributed

Linker


MyProg.exe
0000 main()
110010001…
0010 function11()
10101101…
0030 function12()
1001001…
0080 function21()
10101101…
0090 function22()
1001001…
0110 function31()
10101101…
static linking
objectFile1
0000 function11()
10101101…
0020 function12()
1001001…
objectFile3
0000 function31()
10101101…
0100 main()
110010001…
objectFilei2
0000 function21()
10101101…
0010 function22()
1001001…

Linker


MyProg.exe
0000 main()
110010001…
0010 function11()
search in objectFile1.dll
0020 function12()
search in objectFile1.dll
0030 function21()
search in objectFile2.dll
0040 function22()
search in objectFile2.dll
0050 function31()
search in objectFile3.dll
Dynamic linking
objectFile1.dll
0000 function11()
10101101…
0020 function12()
1001001…
objectFile2.dll
0000 function21()
10101101…
0010 function22()
1001001…
objectFile3.dll
0000 function31()
10101101…
0100 main()
110010001…

Comparison

Static linking

- Easier
- One big file
- New library version → whole file has to be regenerated
- No problems with different versions

Dynamic linking

- Complicated, is used differently on different OS
- Smaller files
- New library version → only library has to be regenerated
- Dll hell!
- Plugins can be realized
- Dll does not become part of your program

Using a library

- Include the header
- Tell the linker which files to include
- Add the path to the headers and libraries
- For dynamic linking make sure that the application can find the dll

Outline

- Organization
- Introduction to depth cameras and Kinect
- C++ basics
- **Assignment**

Installing

- Install OpenNI Unstable 1.1.0.41 Development Edition
- Install avin2/SensorKinect: you might have to manually point Windows to the location of the files when connecting the Kinect
- Install NITE Unstable 1.3.1.5 Development Edition
- Try the OpenNI NIViewer example
- You can use NIViewer to save offline data

Developing

- VS 2008 / VS 9 32-bit solutions are provided
- Should work with VS 2010 / VS10
- Use 32-bit