Intelligent Mirror for Augmented Fitting Room Using Kinect Cloth Simulation

Ngai Ka Kit (2008539150)
# Table of Contents

## Introduction to the project
- Project Overview  P.3
- Tasks  P.3
- Research Findings  P.3
- Approach for cloth-fitting  P.4
- Introduction to Kinect  P.5

## Cloth Simulation
- Cloth Modeling  P.7
- Cloth-Force Structure  P.7
- Cloth Drawing Structure  P.8
- Vector3 Class Design  P.8
- Ordinary Differential Equation (ODE)  P.9
- Forces  P.9

## Interaction between Human and Cloth
- Data Structure of different solids  P.10
- Interaction Logic  P.11

## Mid-term conclusion and evaluation
- Improve performance  P.12
- Change from Object-Oriented-Programming to Procedure Programming  P.13
- Compute Unified Device Architecture (CUDA)  P.14
- Communication between Kinect and C#  P.19

## Conclusion
- Reference  P.21

## Reference
-  P.22
Introduction

Project Overview

With the rapid growth of technology development, our daily lives are heavily affected. Nowadays, people are getting more used to online shopping, online auctions etc to purchase their interested products. This way of transaction has become the main trend and it does bring great convenience to customers. However, an issue for buying clothes online is that you can’t try on them before you get the product. The feeling after dressing on does affect your decision on buying the clothes. Therefore, a kind of virtual “fitting room” can be developed to simulate the visualization of dressing.

The objective of this project is to develop an intelligent mirror that can augment the image of a customer standing in front of it with different clothes fitted to his/her body. In particular, the customer can pose freely in front of the mirror, e.g., turning around to look at his/her back and side-view, and the fitted clothes will keep aligned with his/her body poses in real-time. Such an intelligent mirror can be deployed in the fitting room or advertisement of latest fashion of a garment shop, or even at the home of a customer for shopping over the Internet.

Tasks

1. To develop 3D models for clothes and human body with precise mechanical control
2. To develop a user-friendly application for users to try out different types of clothes for visualization using Kinect
3. To develop possible templates for users to design their own clothes

This report will mainly focus on the first task to develop 3D models for cloth and the calculation of interactive with the human body.

Research Findings

Before we decide the approach, we have done a lot of research online to look for relevant materials. There were already people/teams working on similar projects to develop a virtual fitting room using Kinect. However, most of them are taking the approach to map a 2D texture to the user’s body. Hence when the user moves around you can easily see that the clothes not accurately capturing the user’s position and movement. In order to achieve a more realistic simulation on the clothes fitting process, we have decided to take an approach that requires the construction of two 3D models.
Approach for cloth-fitting

Firstly, we will create a 3D human model according to the user’s dimensions (the body shape, height, width, length of limbs etc.) from the data captured by Kinect. The whole human model will always follow the motion of the skeleton model captured by the Kinect. That is, how the user moves will be reflected by the skeleton model, and our human model will do the same movements.

Secondly, a cloth model will be created according to some mechanics such as frictional forces, gravity, elasticity etc. The cloth model is used to build various types of clothes for dressing on.

In real-time, the interaction takes place between the human model and cloth model. The clothes will be fitted on the human model, and will move in the same way as the human model moves. Hence, a realistic simulation on fitting is done by the interaction between our two models.
Introduction to Kinect

General component
The components of Kinect for Windows are mainly the following:

1. Kinect hardware: Including the Kinect sensor and the USB hub, through which the sensor is connected to the computer;
2. Microsoft Kinect drivers: Windows 7 drivers for the Kinect sensor;
3. NUI API: core of the Kinect for the set of Windows API, supports fundamental image and device management features like access to the Kinect sensors that are connected to the computer, access to image and depth data streams from the Kinect image sensors and delivery of a processed version of image and depth data to support skeletal tracking.

Hardware and streams provided

Fig 1.1 Hardware and software interaction with an application

Kinect sensor mainly provides 3 streams: Image stream, depth stream and audio stream, with detected range from 1.2 to 3.5 meters. At this stage, the first two streams would be utilized for development of human model, cloth simulation and GUI.

The middle camera is a 640×480 pixels @ 30 Hz RGB camera, providing image stream which is delivered as a succession of still-image frames for the application. The quality level of color image determines how quickly data is transferred from the Kinect sensor array to the PC which is easy for us to optimize the program on different platform. The available color formats determine whether the image data that is returned to our application code is encoded as RGB.

The leftmost one is the IR light source with a corresponding 640×480 pixels @ 30 Hz IR depth-finding camera with standard CMOS sensor on the right, which mainly provide the depth data stream. This stream provides frames in which the high 13 bits of each pixel give the distance, in millimeters, to the nearest object at that particular x and y coordinate in the depth sensor’s field of view.
Nui skeleton API

Fig 1.2 Skeleton joint positions relative to the human body

Among NUI API, NUI Skeleton API provides information about the location of users standing in front of the Kinect sensor array, with detailed position and orientation information. Those data are provided to application code as a set of 20 point, namely skeleton position. This skeleton represents a user’s current position and pose. Our applications can therefore utilize the skeleton data for measurement of different dimension of users’ part and control for GUI. Skeleton data are retrieved as aforementioned image retrieval method: calling a frame retrieval method and passing a buffer while our application can then use an event model by hooking an event to an event handler in order to capture the frame when a new frame of skeleton data is ready.
Cloth Simulation

Cloth modeling
A cloth is divided into number of particles, and the following cloth-force structure is used on each edge to join and maintain the particles. The position of the particles will be used to represent the shape of the cloth. In each frame, movement of each particle is calculated by applying physical equations to it. The cloth can then be drawn by the triangle meshes in the following cloth drawing structure.

Cloth-Force Structure
A cloth is formed by the particles. These particles contain edges between them as the forces to maintain the structure of the cloth.

Particle is the core component to simulate the cloth. Each particle contains a number of parameters includes:

1. Mass (float) (get by the area it covers);
2. Position (Vector3) in 3D coordinates;
3. Velocity (Vector3) of that particle moving in one frame;
4. Force (Vector3) applies to that particle in one frame;
5. Color (Color3) defines the color of that particle;
6. Normal (Vector3) for calculating the normal vector of that particle. The average of the normals of its connected plane is required for calculation.

Force edges are used to connect each particles pairs. For each pair of particles, it is required to calculate the attraction force and elastic forces to hold up the structure of the particles.
Cloth Drawing Structure

![Cloth Drawing Structure](image)

The above picture is the cloth drawing structure for visualization of the cloth. The cloth is drawn using Blinn–Phong shading model (in WPF 3D) based on the position and normal (set in the parameter of three peaks of each triangle).

Vector3 Class

This class includes 3 float parameters x, y, z, to represent the coordinates in the 3D geometry. It also contains the operators and common useful functions for 3D vector calculation: i.e.

addition (+), subtraction (-), multiplication (*), division(/), isEqual (==), NotEqual (!=), dot product, cross product, normalize, length, rotation and distance between vectors.
Ordinary Differential Equation (ODE)
For calculating the velocity and the force of the particles in each frame, differentiation is required. The following are the 3 equations that were chosen for the calculation of differentiations.

**Euler Method**

\[ y_{n} = y_{n} + hf(t_{n}, y_{n}) \]

**Heun’s method**

\[ y'_{n+1} = y_{n} + hf(t_{n}, y_{n}), \quad y_{n+1} = y_{n} + \frac{h}{2}(f(t_{n}, y_{n}) + f(t_{n+1}, y'_{n+1})) \]

**Runge–Kutta method**

\[ y_{n+1} = \frac{1}{6}(k_{1} + 2k_{2} + 2k_{3} + k_{4}) \]

\[ k_{1} = hf(t_{n} + y_{n}) \]

\[ k_{2} = hf(t_{n} + \frac{1}{2}h, y_{n} + \frac{1}{2}k_{1}) \]

\[ k_{3} = hf(t_{n} + \frac{1}{2}h, y_{n} + \frac{1}{2}k_{2}) \]

\[ k_{4} = hf(t_{n} + h, y_{n} + k_{3}) \]

After the testing on the three differentiation method, only Runge-Kutta method is suitable for the program. The other two methods have the relatively big error range of the result and will make the cloth become unstable.

**Forces**

**Gravitational-force**
This force acts as the agent that gives weight to the particles with mass and causes them to fall to the ground when dropped. Gravitational-force applied onto each particle is -9.8 in y-coordinates.

**Elastic-force**
This force acts as the agent that brings particles back to its original shape after the stress (e.g. external forces) is removed.

\[ F = -kx \]

is the Hooke’s law where \( x \) is the displacement of the spring’s end from its equilibrium position, \( k \) is a constant called the rate or spring constant.

**Damping Force**
The amplitude of oscillation of the simple pendulum will gradually decrease and become zero when the oscillation stops. The decrease of in the amplitude of an oscillating system is called damping.
Interaction between Human and Cloth

Reasons to use solids instead of the traditional 3D human model:

1. To provide a relatively smooth human model than traditional one;
2. Only require few parameters to represent the solid and provide necessary information (e.g. normal and intersection point) by simple calculation;
3. Better Performance in calculation time;
4. Easy to update the human model when the user moves.

Data structure of different solids

For each cloth particle, it is required to calculate the Interaction between the solid and the cloth particles. The next page gives the details of solid data structure used in building human model.
Interaction Logic

Basically 4 steps:

For each cloth particle, do:

1. Detect if the particle is inside the solid or not, if inside, go to step 2;
2. Calculate the closest point (intersection point) of the particle to the solid surface, set the position of the particle to that point;
3. Calculate the normal of the new position on the surface of the solid, as the direction of the reaction force;
4. Cancel the force to the original force of the particle, apply the frictional force to the inverse direction of the remaining force.

### Interaction Logic for Cylinder solid

```csharp
public ReflectionInfo Interaction(Vector3 V)
{
    ReflectionInfo info = new ReflectionInfo();
    Vector3 A = V - _Centre;
    Vector3 N = -(A.Cross(_Direction)).Cross(_Direction).Normalize();
    float r = A.Dot(N);

    // Step1: Detect if the particle is inside the solid
    if (Math.Abs(A.Dot(_Direction)) > _Length || Math.Abs(r) > _Radius)
    {
        info.Interaction = false;
    }
    else
    {
        info.Interaction = true;
        // Step2: Calculate the closest point to the solid surface
        info.InteractionPosition = V + (_Radius - r) * N;
        // Step3: Calculate the normal of the new particle position
        info.ReflectionNormal = N;
    }
    return info;
}
```
Mid-term conclusion and evaluation

Up to this stage, we have implemented a simple, user-friendly GUI, with a set of controls done by capturing motions of users through Kinect. We have used a combination of 3D solids to construct a fair enough human model, and also the cloth model with some physics mechanics theory on it, together doing interaction with each other to successfully simulate a cloth fitting on the user-body, and visualize it on the screen.

The combine version of the program usefully put the cloth on to a simple human model, Even though the calculation of the cloth is work and can simulate the cloth, but the performance is not desirable. There are two reason lead the program not well.

1. The simulation of the cloth need a huge amount of calculation, the performance of the program become very slow and the frame ratio
2. Because the calculation of the human model objects is also require large amount of calculation. So in current phase, human model can’t use too many objects to build up and it makes the model rouge and it cannot show the detail body shape of the people.

These two problems are directly link to the performance issue of the cloth simulation program. The coming problem is how to improve the performance of the cloth simulation.

At the next stage, we’ll be continuing our focus on fine-tuning our two models. Both accuracy and performance are to be considered to reach an acceptable standard. Improvements may include:

1. use of GPU to increase the speed of calculation
2. Blending force included to improve reality of cloth
3. Find an alternative for 3D drawing, as WPF is not very good
4. Update the Kinect SDK beta from v1.0 to v2.0

Besides, more functionalities will be added to make the software product more complete. The design of the GUI will be adjusted according to users’ evaluations conducted with voluntary participants as well. Cloth design functionality for users will be implemented depending on the progress.
Improve the performance

Change From Object-Oriented Programming to Procedure Programming

Object-Oriented Programming has its advantage, the program can build in a well-defined structure and the objects are easier to handle. But in another side, create, access and delete objects have to use lot of resources from the computer, and it directly affect the performance of the cloth simulation. To improve the performance, change the program from object oriented Programming to Procedure-Programming is necessary, and most class in the program have to cut off.

These are the things changed:

1. Change the vector3 from class to struct. Vector3 class is mainly used in calculation, creating an object every time in the calculation will waste the resources.
2. Remove most of the objects, instead use the array for save the variables. Such as for the particle class, instead the particle contain the properties inside it and then save the particle object into a collection. use the array to save the properties of particles directly.
3. If there are necessary to create the object (such as the 3D drawing function). Try to reuse the object that by changing the value in it, instead of delete old object and add the new object.

Here is an example comparison of code before change and after change.

<table>
<thead>
<tr>
<th>Object-Oriented Programming</th>
<th>Procedure Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class Particle {</td>
<td>public class Particles</td>
</tr>
<tr>
<td>public Vector3 position;</td>
<td>public Vector3[] position;</td>
</tr>
<tr>
<td>public Vector3 velocity;</td>
<td>public Vector3[] velocity;</td>
</tr>
<tr>
<td>protected Vector3 force;</td>
<td>public Vector3[] force;</td>
</tr>
<tr>
<td>protected float mass;</td>
<td>public float[] mass;</td>
</tr>
<tr>
<td>protected float age;</td>
<td>public float[] age;</td>
</tr>
<tr>
<td>protected bool dead;</td>
<td>public bool[] dead;</td>
</tr>
<tr>
<td>protected bool fixe;</td>
<td>public bool[] fixe;</td>
</tr>
<tr>
<td>public Collection&lt;Plane&gt; ConnectedPlane;</td>
<td>public Vector3[] Normal;</td>
</tr>
<tr>
<td>public Collection&lt;Particle&gt; Neighbours;</td>
<td>public uint[] ConnectedPlaneID;</td>
</tr>
<tr>
<td>public Collection&lt;Edge&gt; ConnectedEdges;</td>
<td>public uint[] ConnectedPlaneCounter;</td>
</tr>
<tr>
<td>public Vector3 Normal;</td>
<td>public uint [], NeighboursID;</td>
</tr>
<tr>
<td>public Reaction reaction;</td>
<td>public uint [] NeighboursCount;</td>
</tr>
<tr>
<td>public Friction friction;</td>
<td>public Reaction reaction;</td>
</tr>
<tr>
<td>}</td>
<td>public uint counter, Size;</td>
</tr>
</tbody>
</table>
Compute Unified Device Architecture (CUDA)

Changing the program to the Procedure Programming structure has improved the performance, but it is not good enough to do the real time simulation. It didn’t solve the basic problem of the simulation – the huge amount of calculation on the cloth simulation.

The algorithm of the cloth simulation is obvious, which is directly applying the physical rules to do the calculation. It is difficult to improve the time complexity of the algorithm.

So the next way that I think out to improve the performance is the parallel computing becomes. And CUDA is the solution I choose for the parallel computing.

**Compute Unified Device Architecture (CUDA)** is a parallel computing architecture developed by Nvidia for graphics processing. CUDA is the computing engine in Nvidia graphics processing units (GPUs) that is accessible to software developers through variants of industry standard programming languages, include C++ in visual studio 2008.

Before I try to work on CUDA, I found a testing program to test the ability of CUDA to see is it really useful in improve the performance, and here is the result.

The result show that the calculation speeds of the GPU using CUDA is 16 times faster than the CPU calculation. There and the error range of the result is acceptable in the calculation.
CPU & GPU
In CUDA definition, Host = CPU, device = GPU, and Kernel function is the function that run in device. Parallel portions of an application are executed on device as kernels, one kernel is executed at a time, and many thread in a block and many blocks execute in each kernel.

CUDA is base on the C++ language, but it need to define the variable and functions clearly. Two set of memory device and two processors, we need to define clearly that the code is depends on CPU or GPU.

Functions
- Kernel function - the function which run on GPU, define by __Device__ and __Global__.
- Host function - the function on CPU, same as the normal C++ function, define by __Host__.

Note that only CPU functions can call the GPU functions, but the GPU function cannot access the CPU function. Here is the table to should that the relationship between them:

<table>
<thead>
<tr>
<th>Caller \ Function</th>
<th><strong>Host</strong></th>
<th><strong>Global</strong></th>
<th><strong>Device</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>O</td>
<td>O</td>
<td>X</td>
</tr>
<tr>
<td><strong>Global</strong></td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td><strong>Device</strong></td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
Variable
- Host Variable - save in main memory, normal memory in C++ program.
- Device Variable - the global variable save in the GPU memory.
Local Variable in Kernel Function - which is save in the register in each thread.

<table>
<thead>
<tr>
<th>Caller \ Variable</th>
<th>Host variable</th>
<th>Device Variable</th>
<th>Shared Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Function</td>
<td>O</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Kernel Function</td>
<td>X</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

The Host Variable is completely separate to the kernel function kernel can only get the data from host is passing by parameter when calling kernel function

Other Useful functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CudaMalloc()</td>
<td>Allocate the GPU memory</td>
</tr>
<tr>
<td>CudaMemcpy()</td>
<td>Copying memory function, have three types of copying function.</td>
</tr>
<tr>
<td></td>
<td>cudaMemcopyHostToDevice - copy from Main memory to GPU Memory</td>
</tr>
<tr>
<td></td>
<td>cudaMemcopyDeviceToDevice - copy from GPU Memory to GPU Memory</td>
</tr>
<tr>
<td></td>
<td>cudaMemcopyDeviceToHost - copy from GPU Memory to Main Memory</td>
</tr>
<tr>
<td>cudaFree()</td>
<td>release the memory on GPU</td>
</tr>
<tr>
<td>cudaThreadSynchronize()</td>
<td>CPU function and GPU kernel function are running asynchronies. It is</td>
</tr>
<tr>
<td></td>
<td>necessary to pause the CPU thread for waiting the kernel function finish.</td>
</tr>
<tr>
<td></td>
<td>In another side, pausing the CPU thread will affect the performance of the</td>
</tr>
<tr>
<td></td>
<td>main program. So for the part that don't have data conflict, CPU and GPU can</td>
</tr>
<tr>
<td></td>
<td>run in asynchronies.</td>
</tr>
</tbody>
</table>
Apply CUDA on Cloth Simulation.
For applying the force on each particle, all the data are independent. It can directly apply into CUDA with every thread process one particle. And this can make the calculation of the gravity, damping, velocity, force become parallel.

But for edge force (spring force), it cannot simply apply on the parallel. The edge need to access the data to the two particles it connected, and also need to modify them, so it may have data hazard occurs. Such as following, the order of data access on the force data of particle P.

| EdgeA get the force data of Particle P |
| EdgeB get the force data of Particle P |
| EdgeA set the new spring force to the force data of Particle P |
| EdgeB set the new spring force to the force data of Particle P |

So you can see that the result of calculation of spring force in EdgeA was loss. So for the spring force on edge it cannot simply calculate.

To solve this problem, a pre-calculation is necessary. The edges will divides into few groups, in each group, the edge will not access same particle. And each group of edges will process parallel in one kernel.

So it may prevent the data hazard occurs. The number of group is equal to the maximum degree of the particles.
Here is the data flow between the main program and the CUDA Program.
Communication between CUDA and C#

Create DLL

CUDA is base on the C++ and is only possible to code in Visual Studio 2008 at this moment.

But the Kinect SDK of Microsoft is base on the C#. So how to combine the C# and C++ was became a problem to face.

To build up the communication of C++ and C#, we need to create the .dll library of the CUDA functions and the C# will use the library of the .dll to process the data by define the function again in the C#.

Function in C++

```c++
extern "C" void __declspec(dllexport) __stdcall get(
    CudaVector * __particle_position,
    CudaVector * __particle_normal
){ ... }
```

Define function in C#

```csharp
[DllImport("KUDALIB.dll", CharSet = CharSet.Ansi, SetLastError = true, CallingConvention = CallingConvention.StdCall)]
    public static extern void get(Vector3* particle_position, Vector3* particle_normal);
```

After C# define the function in the header, the C# can call the function normally.

Maximize increase the performance = Minimize the number of calling the GPU

The GPU calculation is fast, but the communication between the CPU and the GPU is still take time. To get the maximized performance of GPU, the amount of copying data between GPU memories must be minimized. In the Cloth simulation, all data of the cloth model will save in the GPU memory. And for every time of update the 3D image, the main program will only copy the position of the particle and the normal vector of particle main memory. This two data are the only data for drawing cloth the 3D cloth to the screen.

Pointer - Unsafe code in C#

C# is a well define programming, and it is work in the object-oriented programming. all data are well manage by the Garbage collection, and it is not allow user to access the address of object by pointer. But it contradicts with the calling function to calling CUDA since that it must pass the data by pointer.

To solve this, it has to set the property of the main program and the compiler to unsafe for access the pointer of the memory.
Memory Allocation
As I mention before, it is necessary for copying the memory from CPU to GPU, but the function calling can only passing the pointer or variable. The data structure of the main program and the CUDA program must be except same, so it can copy the data from CPU to GPU directly. By define the pointer of the first element in the array, the size of array and the size of each value in the array, it can copy all data in the array from CPU to GPU, or reverse. Here is an example of the structure of the Vector structure I create in both C# and C++.

### Memory structure of the Vector3 array in C#

Float (4 bytes each)


### Memory structure of the CudaVector array in C++

Float (4 bytes each)


Garbage collection problem
When the main program calls the CUDA program, it passes a single variable or a pointer to the CUDA program. At the beginning, I don’t want to waste any memory space. I directly point the address of the CUDA program to the address of main program. But it is not workable. Because even the CUDA program have to continue use the memory in those address, but the Garbage collection of the main program will never know that the memory are still in using. It automatically treats it as the useless memory and releases those memories. The memories lose and the CUDA program can’t get the data it need. So not only the memory copy from CPU to GPU, the memory use in CUDA host function, are also need to copy to prevent the Garbage collection delete it.
Conclusion

After Applying the Cloth simulation model with the improved performance, this application has become a complete and acceptable application to provide a virtual fitting room for user to utilize. With the assistance of our group mates, our application has:

1. A delicate human model generated according to user body including shape and dimensions;
2. An impressive, flexible and look-real cloth model for user to “wear”;
3. A easy control, user-friendly and fashionable body-motion-based GUI for user utilized;
4. A lot of interesting and useful functionalities for user to use in our application.
5. 
Reference

**Cloth Modeling:**

http://www.maxgarber.com/projects/cloth/ Real-Time Cloth Simulation

http://www.paulsprojects.net/opengl/cloth/cloth.html OpenGL Cloth Simulation: Balls & Springs - Paul's Project

http://www.jrc313.com/processing/cloth/index.html JRC313.com: Cloth Simulation:

**Geometry Calculation:**

http://mathworld.wolfram.com/EllipticCylinder.html Elliptic Cylinder

http://www.math.hmc.edu/~gu/curves_and_surfaces/surfaces/ellipsoid.html The Ellipsoid

http://i.cs.hku.hk/~c0271/ CSIS0271 Computer Graphic Homepage

Physical momentum:

http://regentsprep.org/regents/physics/phys01/friction/default.htm The Force of Friction

**Cuda:**

http://people.maths.ox.ac.uk/~gilesm/hpc/NVIDIA/NVIDIA_CUDA_Tutorial_No_NDA_Apr08.pdf Tutorial CUDA

http://www.nvidia.com/content/cudazone/download/Getting_Started_w_CUDA_Training_NVISION08.pdf Getting Started with CUDA