Department of Computer Science
The University of Hong Kong

CSIS0801 – Final Year Project

Detailed Interim Report
Intelligent Mirror for Augmented Fitting Room Using Kinect

Supervisor: Dr. Kenneth Wong
2nd Examiner: Dr. Anthony Tam
Members: Ho Chung Yin, Billy 2008196829
         Ma Cheuk Kiu, Charles 2008303898
         Ngai Ka Kit, Kit 2008539150

Dated as: 9th January, 2012
# Table of Contents

Table of Contents ................................................. 2
Project Overview .................................................. 3
Tasks ................................................................. 3
Research Findings .................................................. 4
Approach for Cloth Fitting ........................................ 4
Introduction to Kinect ............................................. 5
  1. General components ....................................... 5
  2. Hardware and Streams provided ......................... 5
  3. Nui Skeleton API .......................................... 6
Graphical User Interface (GUI) Design ....................... 7
  1. Phase 1 ....................................................... 7
  2. Phase 2 ....................................................... 9
Human Body Modeling ............................................ 12
  1. 1st Approach: Blender + XAML exporter ............... 12
  2. 2nd Approach: Combination of simple 3D solids .... 16
  3. Improvements .............................................. 18
Cloth Modeling .................................................... 19
  1. Cloth-Force Structure .................................... 19
  2. Cloth Drawing Structure .................................. 20
  3. Vector3 Class Design ..................................... 20
  4. Ordinary Differential Equation (ODE) ................. 20
  5. Forces ....................................................... 21
Interaction between Human and Cloth ....................... 21
  1. Data Structure of different solids ...................... 22
  2. Interaction Logic ......................................... 23
Functionalities .................................................... 24
Result on current version ...................................... 25
Conclusion .......................................................... 26
References .......................................................... 27
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
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**

1. **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.

2. **Hardware and streams provided**

   ![Diagram](image)

   **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.

3. Nui skeleton API

![Fig 1.2 Skeleton joint positions relative to the human body](image)

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.
GUI Design

With the Microsoft Kinect device as a motion sensing input device, we can develop a GUI controlled by user motions in order to provide an interactive environment for better user experience.

**Phase 1:**

Human motion tracking would be our first target to be finished. By comparing different software development kit for Kinect released on network, our group have decided the official Kinect SDK - The Kinect for Windows SDK Beta v1.0 (latest version at that time) as our development platform as this seems to have more concrete support by Microsoft other than normal programmers and Kinect hackers.

Focusing on detecting user motion, the essential part of Kinect to be utilized is Skeletal tracking, i.e. to track the skeleton image of user moving within the Kinect field of view. It allows us to create gesture-driven applications.

**Implementation:**

For a WPF project, XAML is main component for creating user interfaces with code-behind in, for example using C#, xaml.cs. In this phase, skeleton stream is used for capturing user action and video stream for providing real-time image of user. For each stream an event handler is implemented to handle event changes. Smoothing on skeleton engine is also added for retrieving a more steady skeleton.

For *Skeletons* in each skeleton frame, *SkeletonData* is stored inside which provides different information of the particular *Skeletons* at *SkeletonFrame*. This includes: *Joints, Position, Quality, TrackingID, TrackingState* and *UserIndex*.

There are 3 states pre-set:

- **Tracked** – indicates success of capturing the *Skeleton*;
- **Position only** – indicates some part of *Skeleton* is chopped, out of detection field by Kinect or not 100% confirmed to be correct;
- **NotTracked** – indicates that *Skeleton* has not been tracked.
For correct skeleton tracking, the TrackingState of Skeleton should always be ensured as Tracked, and the user’s motions will be handled.

In this phase, skeleton is drawn out for testing purpose. It includes mainly two parts: bones and joints. By coloring with Brush using a set of colors for different parts of skeleton (i.e. body segment), they are drawn and added to the Skeleton. Parts like left arm, right arm, left leg, right leg and (head to hip) are individually drawn using Polyline. For each part of Skeleton, joints are drawn by points, which are pre-defined with different colors in order to identify the joints.

```csharp
private void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
{
    SkeletonFrame skeletonFrame = e.SkeletonFrame;
    Brush[] brushes = new Brush[6];
    brushes[0] = new SolidColorBrush(Color.FromRgb(255, 0, 0)); ...

    skeleton.Children.Clear();

    foreach (SkeletonData data in skeletonFrame.Skeletons)
    {
        if (SkeletonTrackingState.Tracked == data.TrackingState)
        {
            // Drawing bones
            Brush brush = brushes[iSkeleton % brushes.Length];
            skeleton.Children.Add(GetBodySegment(data.Joints, brush,
                JointID.HipCenter, JointID.Spine, JointID.ShoulderCenter, JointID.Head)); ...

            // Drawing joints
            foreach (Joint joint in data.Joints)
            {
                Point jointPos = GetDisplayPosition(joint);
                Line jointLine = new Line();
                jointLine.X1 = jointPos.X - 3;
                jointLine.X2 = jointLine.X1 + 6;
                jointLine.Y1 = jointLine.Y2 = jointPos.Y;
                jointLine.Stroke = jointColors[joint.ID];
                jointLine.StrokeThickness = 6;
                skeleton.Children.Add(jointLine);
                SetEllipse(joint.ID, jointPos, joint);
            }
        }
    }
}
```
Phase 2:

In this phase, there are 3 major objectives on user interface:

1. Graphics – design and style should be nice-looking to users;
2. Easy Control – the control method should be intuitive, e.g. users use their hands to touch the virtual buttons to access different functions;
3. User-friendliness – the location of buttons and menus should be well-presented for users’ convenience;

For graphics, Photoshop is used in order to design dedicated and pretty user interface with different effects. The whole design style mainly follows the rule “Simple is the best”.

For easy control, skeleton stream retrieved from Kinect can be utilized for motion capturing. The user’s hand will be shown on the screen, where left and right hand are represented by red and green hand icon respectively. Users use their hands to control buttons as the same way cursors do on GUI.
For user-friendliness, it is brought by intuitive graphical controls in which users can easily understand what the control refers to at their first sight. In traditional user interface, confirmation of selecting items and using functions requires clicking by mouse on a confirm button. This is not applicable in our GUI design as users would feel inconvenient to repeatedly move their hands onto same button in order to confirm. Therefore, we use the concept of hover button: users can confirm by ‘touching’ the item or button for few seconds with their hands for completing the action. This can also avoid the user touching a wrong control button when they move their hands across the interface. Users’ hand movement is also captured for implementation of a scrollbar. By moving the hand icon from top to bottom, user can browse the next page of the clothing catalog.

```csharp
private bool CheckTakePhoto()
{
    double leftX = Canvas.GetLeft(handRightEllipse) + 25;
    double leftY = Canvas.GetTop(handRightEllipse) + 25;

    if (leftX < 490 && leftX > 455 && leftY < 70 && leftY > 30)
    {
        if (!t3.Enabled)
        {
            t3.Enabled = true;
            t3.Elapsed += new System.Timers.ElapsedEventHandler(t_TakePhotoTimeOut);
            handRightEllipse.Hovering();
        }
        return true;
    // remaining...
```
The picture below explains the GUI created in this phase:

![GUI diagram](image)

Fig 1.5 Tutorial page

The circular rotation disc in the top-left corner is the category button. Users can change the category shown among clothes, trousers, shoes and other accessories. The white round-cornered rectangle with scroll bar on the right is the catalog panel showing the items in selected category. Users can select their want-to-try item by the green hover button. As mentioned, the scroll bar is used to navigate through pages of the catalog.

The three buttons on the top-righthand corner are the functionalities we have completed in this phase to provide users with better user experience.

The leftmost button is the photo-taking button. Users can take photos with the chosen clothes dressed on their bodies. The screen will show a count from 5 to 0 for users to take their poses. Photo will then be saved in the folder created by the program and users can retrieve the photos there. The middle button is trash button which allows users to clear what they have worn. The rightmost button is tutorial button. Users who want to look at the instruction of this application can choose this function, and the tutorial page in Fig1.4 will be shown on the screen.
Human Body Modeling

Normally, a detailed 3D human model is constructed by a mesh geometry object formed by a lot of triangles or quadrilaterals. The things that you have to give are the vertices, edges, normals etc. However, several difficulties arise.

1. The structure of a human body is so complicated that it’s not easy to draw the whole model by hand, as you have to define the positions of every vertices, and the design of the whole mesh;
2. The data captured by Kinect is also very limited. With the skeleton model giving you the 20 points, and the depth map giving you the distance information, it’s hard to produce a full image of the user’s body;
3. The time complexity may be a problem. To get a more precise model, it’s preferred to have more number of triangles. However, this may slow down the interaction process between the two models and downgrade the overall performance.

Therefore, we have investigated 2 approaches for opting a feasible solution. The first one is to look for existing software to make use of those free 3D models available online. The second one

1st Approach: Blender + XAML exporter

Since there are actually many free 3D models available in online resources, this approach tries to find existing softwares to make use of these models to construct our own human body model (cloth model as well) by doing some conversions to fit on our data structures.

Introduction to Blender

Blender is a free and open-source 3D computer graphics software product used for creating animated films, visual effects, interactive 3D applications or video games. It supports a lot of different formats for 3D objects, such as .obj, .3ds, .max etc. Therefore, we can import the 3D models downloaded into blender very easily. Also, provided with the editing tools we can do adjustments like cutting to polish the 3D model according to our necessities.
Introduction to XAML
XAML (Extensible Application Markup Language) is a part of WPF. It is a declarative XML-based language created by Microsoft used for initializing structured values and objects. By using XAML, we can easily define our 2D/3D objects in the space by modifying the corresponding attributes. For example, for a mesh geometry object, what we need is the definition of its vertices, directions, normals. The code in .xaml will be as follows:

XAML exporter
This is an extra add-on to Blender. After you imported the 3D models downloaded, you can export the model into .xaml file directly. The exporter will do all the conversions for you automatically. (All definition of vertices, lighting, texture etc – it exports the whole scene.) After getting the .xaml file, we can use the data to construct our own model.

Flow for the 1\textsuperscript{st} Approach

- Obtain free 3D models from web
- (both human and clothes)

- Adjust the models (e.g. cutting)
- Export the 3D models to .xaml format

- Change the format to suit our needs
- Adjust attributes to fit dimensions
1\textsuperscript{st} step: Finding the 3D models
There are a lot of 3D models available for free on the Internet. We hope to find a simple human model with less detail (performance issue). Different types of clothes are especially useful for our later phase.

2\textsuperscript{nd} step: Working with Blender
For the human model, we first import the whole model into Blender. Then we cut it into individual parts to obtain (head to neck), (upper arm), (forearm) etc. These individual parts will be mapped to the corresponding positions of the skeleton model captured by Kinect. Hence, for user’s movements we can just do some transformations on the body parts. Each body piece will be exported to .xaml file to be used in the next step.

Fig 2.1 Cutting of the forearm piece
3rd step: Formatting .xaml files
All the information from the .xaml file will be used as input parameters in our program to construct the 3D objects. For example, the design of the Arm is as follows:

```csharp
public class Arm : Body {
    public double length;
    public MeshGeometry3D mesh_arm;
    public Collection<Point3D> Points = new Collection<Point3D>();
    public Collection<int> Orders = new Collection<int>();

    public Arm(double length);
    public void on_render();
}
```

In the main program, the Arm is constructed by giving the length. In the on_render() method, the .xaml file will be read. And the data is stored in the Collection<> objects. Then it will pass the Collections<> to call the method CreateModel() inherited from the parent Class: Body. It then creates the whole MeshGeometry3D for the body part.

```csharp
public MeshGeometry3D CreateModel(Collection<Point3D> points, Collection<int> orders) {
    MeshGeometry3D mesh = new MeshGeometry3D();
    for (int i = 0; i < points.Count; i++)
    {
        mesh.Positions.Add(points[i]);
    }
    for (int j = 0; j < orders.Count - 2; j += 3)
    {
        mesh.TriangleIndices.Add(orders[j]);
        mesh.TriangleIndices.Add(orders[j+1]);
        mesh.TriangleIndices.Add(orders[j+2]);
        Vector3D normal = CalculateNormal(points[orders[j]], points[orders[j+1]], points[orders[j+2]]);
        mesh.Normals.Add(normal);
        mesh.Normals.Add(normal);
        mesh.Normals.Add(normal);
    }
    return mesh;
}
```

However, after some experiments we found that this approach will incur a lot of calculations when doing interaction between cloth model and this human model. (compare huge number of vertices with the clothes..) Hence, the performance is not
satisfactory for a smooth run. We need to find a more feasible solution to do the Human Modeling part.

But still, we can still use Blender as a convenient tool to visualize 3D objects, and to make various clothes using this approach.

**2nd Approach: Combination of simple 3D solids**

The main objective to create the human body model is not to visualize it. It only serves as an in-between process to do the interaction between clothes and user body displayed on screen. So if we can find way to tackle the simulation, we don’t need to actually define every details of the “invisible” human model (and it is indeed invisible).

![Fig 2.2 simple 3D solids](image)

The 2nd approach we use is to construct the model by a combination of different simple 3D solids, such as spheres, plane, cylinders, frustum, ellipsoids etc.

As shown on the diagram above, a simple human model can be constructed by these simple solids. For examples, the arms, legs etc can be modeled by circular cylinders; while head and joints can use simple spheres.
However, you may ask: the model is rough, how can you make realistic modeling of the whole user body? The tiny part can actually be adjusted by some small solids attached on the surface of the larger parts as shown below.

![Fig2.4 Minor adjustments on model](image)

Adjustments can be made very easily by constructing small 3D solids attached to the main body parts. The only issue is how to get the shape correctly and precisely of users captured by the Kinect device. More deep investigations on depth maps will be carried out later to achieve the purpose. Therefore, we now only use these simple models to do the fitting.

## The implementation

The interface `Solid`

```java
public interface Solid
{
    ReflectionInfo Interaction(Vector3 V);
}
```

`ReflectionInfo`

```java
public class ReflectionInfo
{
    public bool Interaction;
    public Vector3 ReflectionNormal;
    public Vector3 InteractionPosition;
}
```

The Solid class is used as an interface for all children solids. Each children class must implement the `Interaction(Vector3 v)` method.
An example: Sphere.cs

```csharp
public class Sphere : Solid
{
    private Vector3 _Centre;
    private float _Radius;
    private float _RadiusSquare;

    public void Update(Vector3 Centre);
    public ReflectionInfo Interaction(Vector3 V);
}
```

There are only 3 variables stored in Sphere.cs – its centre, its radius, and its radius square. These values will be used in the Interaction() method for doing the calculation of interaction between human and cloth model. The Update() method just keep updating the new centre position of the sphere.

The logic for the interaction will be discussed in the physics part.

So, as we can see the amount of data needed to represent the human model becomes very small, the performance issue now depends on the algorithm we take to do the interaction logic between human and cloth model. Hence, we choose this approach as the feasible one.

**Improvements on Human Modeling**

Since this approach is revised at a late stage, the measurement for the model is only a rough estimation using data from skeleton (length between two joint points). In order to construct a more accurate human model with shapes like belly, breasts etc, more research will be done on the Depth Map to investigate how the data can be used to do precise measurements of the user’s body.
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

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, integration is required. The following are the 3 equations that were chosen for the calculation of integrations.

**Euler Method**

\[ y_{n+1} = y_n + hf(t_n, y_n) \]

**Heun's method**

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

**Runge–Kutta method**

\[ k_1 = hf(t_n + y_n) \]
\[ k_2 = hf \left( t_n + \frac{1}{2} h, y_n + \frac{1}{2} k_1 \right) \]
\[ k_3 = hf \left( t_n + \frac{1}{2} h, y_n + \frac{1}{2} k_2 \right) \]
\[ k_4 = hf \left( t_n + h, y_n + k_3 \right) \]
\[ y_{n+1} = \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4) \]
By testing, Euler Method gives the best performance in time complexity but have relatively larger error range in the result. Runge–Kutta method requires more time to calculate but is with the smallest error range among them.

**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*.

**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.
Sphere

Elliptic Cylinder

Plane

Frustum

Ellipsoid
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.

```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);
    if (Math.Abs(A.Dot(_Direction)) > _Length || Math.Abs(r) > _Radius)
    {
        info.Interaction = false;
    }
    else
    {
        info.Interaction = true;
        info.InteractionPosition = V + (_Radius - r) * N;
        info.ReflectionNormal = N;
    }
    return info;
}
```
Functionalities

1. Photo-taking
This functionality enables user to save the screenshot image just like taking a photo. With the clothes dressed on the user’s body, the user can “press” the photo button on the panel. Then, a 5-seconds countdown will be displayed on the screen. Users can make use of the time to make their own pose. The output picture will be automatically saved to:

C:/Users/username/My Documents/KinectFitter/Photos
with the file name IMG_XXXX.jpg where XXXX is the id of the photo.

```
public static void SaveCanvasToFile(Canvas surface, string filename)
{
    Size size = new Size(surface.Width, surface.Height);
    surface.Measure(size);
    surface.Arrange(new Rect(size));

    // Create a render bitmap and push the surface to it
    RenderTargetBitmap renderBitmap = new RenderTargetBitmap(
        (int)size.Width, (int)size.Height, 96d, 96d, PixelFormats.Pbgra32);
    renderBitmap.Render(surface);

    // Create a file stream for saving image
    try
    {
        using (FileStream outStream = new FileStream(filename, FileMode.Create))
        {
            BmpBitmapEncoder encoder = new BmpBitmapEncoder();
            // push the rendered bitmap to it
            encoder.Frames.Add(BitmapFrame.Create(renderBitmap));
            // save the data to the stream
            encoder.Save(outStream);
        }
    }
    catch (DirectoryNotFoundException)
    {
        Directory.CreateDirectory("C:\Users\" + Environment.UserName + "\My Documents\KinectFitter\Photos");
        using (FileStream outStream = new FileStream(filename, FileMode.Create))
        {
            BmpBitmapEncoder encoder = new BmpBitmapEncoder();
            encoder.Frames.Add(BitmapFrame.Create(renderBitmap));
            encoder.Save(outStream);
        }
    }
}
```
Result on current version

The above picture shows the current implementation of our application. After doing the body configuration, user can choose clothes from the catalogue. The testing cloth can be successfully dressed on the user’s body. When the user slightly moves around his/her body, the cloth will do the corresponding movement as well.
Conclusion

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.

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.
References

Blender 3D models + XAML exporter:

http://e2-productions.com/repository/ The Official Blender Model Repository
http://charly-studio.com/blog/blender-2-5-wpf-xaml-exporter/ Blender 2.5 WPF XAML Exporter
http://www.3dcool.net/ Free 3D models repository

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:

GUI

http://channel9.msdn.com/coding4fun/kinec Kinecting the Dots: Adding Buttons to your Kinect Application
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

Kinect SDK:


Physical momentum:

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

Windows Presentation Foundation (WPF)

http://www.wpftutorial.net/IntroductionTo3D.html WPF Tutorial.net
http://kindohm.com/technical/WPF3DTutorial.htm Windows Presentation Foundation (WPF) 3D Tutorial