A Kinect Interface for StoreWorld

Abstract

StoreWorld is a facebook game being developed by students at RIT, as a joint effort between the College of Business and the School of IGM. It is a store-management game similar to Cafe World, but the stores in question are fashion stores rather than restaurants. One of the game’s main features AI agents that behave according to real-world business models to create a realistic economy.

The idea for the Kinect Kiosk is to allow players to control their StoreWorld avatar using the Kinect, and have them try on and buy clothes from other people’s StoreWorld stores.

I developed a prototype of the kiosk, where a user can control the default StoreWorld avatar, and switch between two clothing pieces: a suit coat and a dress shirt. The prototype does not yet interface with StoreWorld itself, but it does use art assets from the game.

Concept

The general idea of the kiosk is to create an alternate way to interact with the StoreWorld world. The Kinect is a great way to attract people’s interest and attention to StoreWorld during special events, since it’s fun and intuitive to play with. It can bring in new users who make an account just to play with the Kinect-controlled avatar and then continue to use StoreWorld at home. At the same time, existing users may be interested in viewing and interacting with the familiar world in a whole new way.

In the original concept of the Kinect Kiosk, the user would be able to log into his facebook/StoreWorld account; he would then see and control his own customized avatar. He would be able to enter his friends’ and other users’ StoreWorld stores and browse their inventories, trying on any clothes he finds interesting. He would then buy any of the clothes he liked, affecting both the store’s inventory and his own closet.

Background Research

There are several “virtual fitting rooms” out there, of varying degrees of quality. They all overlay virtual clothes over the video feed of the real-world person interacting with the software. Some programs use the Kinect to track bodies, while others use simple webcams and their own tracking algorithms. None of the programs I saw dealt with limbs that move independently from the body - the clothing was always one solid piece that did not move relative to itself.
Most of these virtual fitting rooms simply overlay a single flat stored image (photograph) of a clothing item onto the body, and try to keep it in roughly in the right place relative to the user’s body (or they simply ask the user to position the item themselves, by hand). There are some variations on this theme, such as using a 3D model of the clothing item, and rotating it when the actual user’s body rotates, or storing several photos at slightly different angles, and switching between them when needed.

Here are some of the more interesting virtual fitting rooms I looked at:
http://www.youtube.com/watch?v=rn_iPiGKd0M&NR
http://www.youtube.com/watch?v=XiiQ7MXe1-8
http://www.youtube.com/watch?v=yUroQw_ydw0
http://www.youtube.com/watch?v=L_cYKFdP1_0

**Implementation**

In order to start programming the application, I first had to install the AS3 Kinect libraries on my computer. I chose to use OpenNI rather than OpenKinect as the middleware between AS3 and the Kinect, since OpenNI was the only one to offer skeleton data. One drawback to the OpenNI version of AS3 Kinect, though, is that it does not provide the video feed from the Kinect - only depth and skeleton data.

I used these instructions to install AS3 Kinect and the libraries it depends on: [http://www.as3kinect.org/guides/openni-win32-wrapper-guide][1]/http://www.as3kinect.org/guides/openni-win32-wrapper-guide/

I did run into quite a few problems with using the correct versions of all the software, since these instructions are slightly outdated, the versions get updated constantly, and it’s often hard or impossible to find previous versions to download and install.

The trial-and-error process of downloading and installing various versions of the libraries, and restarting the computer at just the right times in the installation process, in various orders was probably the most time-consuming part of this project.

Once everything was installed, I modified the sample code provided to superimpose scaled pieces of the StoreWorld avatar onto the limbs, torso and head of the skeleton provided by the Kinect. I cut up the avatar into pieces and edited each piece so that joining two pieces at any angle (while keeping the pieces in the same plane, with no scaling) would produce relatively smooth joints. I also rotated each piece so that the vector from the “parent” joint (e.g. shoulder for the upper arm) and the “child” joint (elbow for the upper arm) would lie exactly along the y-axis. That way, I could scale the X and Y dimensions of the image independently, and scaling by the Y dimension would produce the effect of foreshortening. Scaling along the X dimension helps adjust the width of the limbs based on how close the avatar is to the screen, and therefore how big he should be overall. The torso was an exception - I scaled the X dimension of the torso to match the shoulder width of the skeleton data.

Once the body itself was done, putting on the clothes was relatively simple - I just had to scale each piece of the clothing art to match the corresponding limb exactly. Theoretically, that
means that the clothing would always be attached to the limb in the exact same way, and it would always cover up the limb if the art itself was adjusted properly (practically, you can make the elbows poke through the suit top, even though the suit art covers up the upper arm art completely.) Switching clothes was simply a matter of attaching and detaching all the clothing pieces to and from the corresponding limbs.

I also implemented a kinect button, which would animate for one second while one of the hands was hovering over it, and then execute whatever function you passed to it. This way, if a user accidentally brushes his hand past the button, it would not get “pushed”, but if he holds his hand over it for a little bit, it does activate. I made two buttons, one each for the two pieces of clothing I could switch in and out.

**Results**

The resulting Kinect application is quite simple, but still very fun to play with. It tends to attract a lot of attention when it is being demonstrated at events like Imagine RIT or the IGM Showcase. People, especially kids, enjoy playing around with the cartoon avatar, and making him do silly things.

Since the algorithm for translating from 3D skeleton to 2D avatar is quite simple, there are still quite a few things that make the avatar look funny and just wrong under certain circumstances. However, that didn’t seem to detract from the enjoyment of playing with the avatar. In fact, the unintended features and bugs seem particularly fun for people to play and experiment with: shrinking the avatar’s head by putting your hands over it, making the head look disconnected from the body, just generally confusing the Kinect about the skeleton position.

As a proof-of concept, I think my application is quite successful.

**Lessons learned**

*Installing hacked libraries on top of other hacked libraries is hard* - It took me far longer to get the AS3 Kinect libraries installed than to actually use them to create the application

*Editing art is hard* - I essentially had to create a paper doll with articulated joints out of a single piece of avatar art. I had to first split up the avatar into limbs, then add “joints” to these limbs in the right places so that rotating the limbs still looked more or less good. It was hard to figure out exactly where to break up the art, and where and how to add the joints.

*Converting 3D point/angle data to 2D avatar representations is an interesting and complicated problem* - the “articulated paper doll” solution isn’t quite robust, since the paper doll would only move in one plane, while in the real world, we move in three dimensions. So, when the joints weren’t all in one plane, the art could look somewhat wrong. There are also issues of rotating the body (my animated avatar always faces the front, and his hands always face sideways, with the thumbs in front) and scaling the width of the limbs correctly.
**The Kinect is not perfect: but using it in better environments make a huge difference**
- the Kinect lost track of the skeleton quite often, but would sometimes act as if it hadn’t lost track at all, refusing to re-calibrate. I had to put in a manual “reset” button and use tricks such as covering up the cameras momentarily to get it to recalibrate. However, using the kinect in a well-lit area with a white background and no other people walking around in the back made a surprising amount of difference: the Kinect was almost perfect with respect to calibration and tracking.

**The Kinect skeleton data has some peculiarities** - The main one is that the shoulders, hips, neck and head act as if they are all fused together. They are always all in the same plane, and in the same relative positions. The one exception is when other limbs come too close to the head - the neck/head data points then stretch and move around, as if to avoid the other limbs. This means that the head of the avatar can shrink, grow, and get separated from the body. The other peculiarity is that the data AS3 Kinect provides seems to be in projected coordinates - the X and Y correspond to the positions of the skeleton points on the screen, while the Z tracks the relative depth of the points (and the Z seems to be measured in different units than X and Y). This works well for projecting the 2D art onto the skeleton, but it does not quite work for measuring the actual physical limb length, since I would need actual 3D coordinates to do so accurately.

**Controlling an avatar with one-to-one physical movement and real-time feedback can be really fun and conducive to free play** - Most of the people who tested out the program enjoyed simply moving around and seeing how the avatar responded. They also liked doing things that would make the avatar look silly - shrinking his head by putting their hands over it, making him flat by turning sideways (in earlier implementations), moving around really fast until the Kinect got confused about where all the limbs actually are and contorted the avatar into funny shapes. I think controlling an avatar with your own body gives users a feeling of entering a new world, becoming someone else, more than controlling a video feed image of themselves would: the avatar works through slightly different rules, so it does feel like a different world, and yet it is similar enough where the user can predict what his movements will do.

**Future Work**

The College of Business and the StoreWorld team has hired me to develop this project further over the summer. I plan on making the 2D avatar algorithms more robust, developing more features, and connecting the application to the StoreWorld game.

The connection to StoreWorld is one of the main features that’s still completely absent: being able to visit other people’s stores, making purchases that persistently affect both your and the store owner’s account, being able to see your customized avatar instead of the generic one. This part of the project depends in part on the StoreWorld development itself, as the necessary pieces (persistent stores, persistently customisable avatars, being able to own clothes) need to be present in StoreWorld in order for my application to connect to them.
I will also need to work with the StoreWorld artists to get more art, and art that is more applicable for this program. Specifically, I will need the avatar (and clothing) art broken up into many pieces, such as joints and limb parts, with many different views from many angles. These pieces need to be able to work together so that the avatar looks “right” in any position, from any angle.

I will need to modify the conversion algorithms to make them more robust: right now, the “limb” pieces that scale to the 2D projection of the skeleton are the only type of 3D to 2D conversion. However, this makes the joints look funny, especially when there is a lot of foreshortening going on - for example, when the arm is stretched forward, so that the arm piece is scaled to be quite short, together with the shoulder part. The shoulder then does is not connected or scaled properly. At the very least, I need to add a “joint” type piece, which will not be affected by the foreshortening of the limbs, and will usually be a near-circular piece (representing a sphere around the joint). Perhaps some use of splines at the joints will also make the whole avatar look smoother.

As part of a better avatar structure, I may need to perform some post-processing on the skeleton data that comes from the kinect. For example, the human body actually has two points of articulation at the shoulder: the shoulder blade and the arm bone itself. The kinect only has one joint at the shoulder, which makes the arms look especially funny if you raise them all the way up. With some simple inverse kinematics between the elbow and shoulder, I may be able to insert a second point of articulation, and make the shoulders look better. Similarly, I could perform some calculations and heuristics on the hips/shoulders/neck/head structure (which acts as if it is all fused together in the kinect data) and make the head be able to move and tilt independently of the body.