Evan Denmark
Chandani Doshi
6.837 Final Project – Microsoft Kinect Motion Capture Graphics – Group 16

Background

In the film industry today, both animation and visual/special effect play an important role in developing realistic and believable shots, giving the audience a sense that the characters and set on screen are not fictionalized, but actually part of the audience's world. In order to create such a sense, filmmakers must focus on one of the most critical aspects of realism in their films: motion. Capturing realistic motions for characters has been around since the rise of animated films when Walt Disney's film studio began taking live action films of actors only to then trace the outline of the cartoon characters over the film frame. Of course, this process began with artists drawing over the films, but over time, this process became integrated with more advanced technological systems.

It wasn't until the mid 1990's that the modern concept of motion capture was truly invented. Some of the first characters to be motion captured with a human actor include Gollum from *Lord of the Rings*, Val Kilmer from *Batman Forever*, as well as large crowds in James Cameron's *Titanic*. The technique used in these films is not radically different than what is used today. Actors will wear specific sensors located at pivotal places on the body. Then, using multiple cameras at different angles, filmmakers will capture the actor's movements in a 3D space. They will then use the sensor information to create a 3D graphical representation of the scene. Then, after an artist's rendition of the character, the scene will be able to be viewed from any virtual camera angle.

In the past twenty years, a significant amount of technological advancement in the graphics and film industry has allowed for motion capture to become a regular part of film making. Motion capture systems have become so good that many films seek to capture realistic facial expressions with the use of small cameras placed in front of the actor's face, giving their character a similar facial expression, further increasing the realism of the film.

Although major films studios still use sensors to capture movements of an actor, recent systems, including the Microsoft Kinect, have found ways to detect a person and track their movements without the need for expensive sensors. The Kinect uses the combination of the color camera and two infrared "depth cameras", which detect how far away an object is from the camera. Because the two depth cameras are spaced far apart, it allowed the systems to sense depth very similar to how the human eyes perceive depth. Of course, there are limitations: the Kinect sensor must be stationary and accuracy is only guaranteed within a certain distance to the camera. However, for being a relatively cheap and publicly available alternative to film studio motion capture, the Microsoft Kinect does a pretty good job of enhancing the user experience in video games and animation.

Motivation

As described above, using motion capture can make the animation of the character extremely realistic, but it also provides a significant amount of aid to the animator. As we

showed in Assignment #2 (the bone hierarchy and skinning pset), it is very difficult to animate a character by adjusting one joint at a time. By using motion capture, you are able to get a very accurate representation of a character in a much easier and more efficient way.

Personally, our motivation for this project was that we wanted to do something bold and by doing something that we had no experience in. Neither of us had worked with the Microsoft Kinect or Maya before, so we wanted to experiment with both. Overall, we learned an incredible amount through this project and are really glad we chose it.

Approach

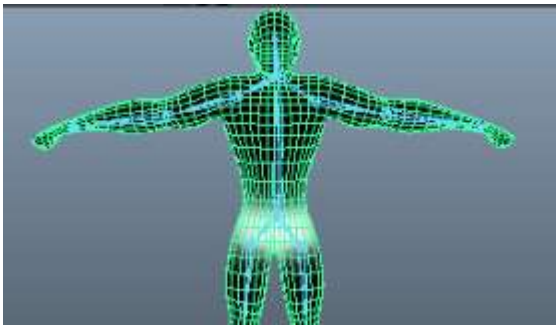
From the beginning, we wanted to do some kind of motion capture animation, but we had many ideas and eventually settled on the Microsoft Kinect. Our initial idea was to model a graphical face based on actual feature based detection using OpenCV. However, when we tried to do facial recognition on OpenCV, we came to the conclusion that it might not be as accurate as we need it to be. OpenCV allows the user to detect very basic facial features (eyes, nose, mouth, profile face, etc…), but does not allow for detail, which would be necessary to create a realistic looking face. In addition, OpenCV would not provide us 3D data, as the Kinect does, but instead it would only provide the x, y coordinates of features on a 2D screen. This would leave a larger problem because we would then have had to infer 3D points from 2D points. Professor Wojciech then suggested using a Kinect sensor to acquire data and augment Assignment 2. Because the Kinect sensor has 1 color camera and 2 depth cameras, the Kinect is able to detect 3D data and then translate that into a skeleton for the user.

Using the Kinect for Windows SDK v1.0, we were able to detect the 20 joints of a human skeleton at 30 frames per second, record them in a CSV file, and then export them to an external program. As mentioned before, we initially wanted this external program to be Assignment #2. Therefore, we decided that the motion capture should not be in real time, but rather recorded because the Kinect was compatible only with Windows and Assignment #2 was on Athena Linux (and it would have been a pain to integrate them).

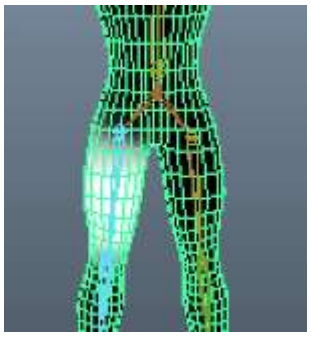
In our first approach to build on Assignment #2, we created a method that read csv file and stored the coordinates of the joints in a vector. Then we wrote a method to update the joint transformation matrix for a given time step using the appropriate angles of rotation. Our joints were being correctly calculated in time, but when we ran our code, we noticed that the model viewer was not being updated at each time step even though the joint transformation matrix was being updated. We believe that this may have been due to thread issues, but the exact reason is still unknown to us. After speaking with Wojciech, he suggested we use an external software to view our figure.

Our next approach was to use Maya, a 3D animation and modeling software. We watched and read multiple tutorials on how to rig a model, and were successfully able to build a skeleton and rig the model. We wrote a Python script to read the csv file and create keyframes at each time step, which animated our figure in time. When imported into Maya, our skeleton joint system was perfect animated. However, we did run into one small problem: our skin was not correctly following our bone structure. Even though we self-rigged our model and hand painted the skinning weight onto the figure, some areas of skin were still not exactly right. As you can see in the below photos, our skinning weights seem

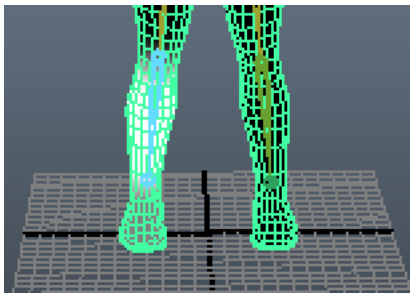
to be correct when in the bind pose and the skeleton bone structure is correct in the final output video (at the bottom). Yet, if you look carefully, you can see that in the arms specifically, there are unwanted curves.



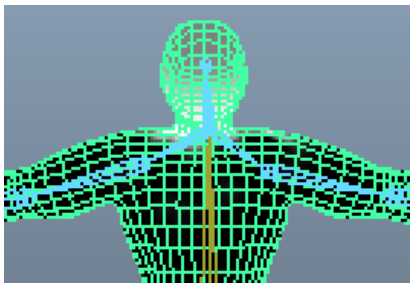
Hip Center



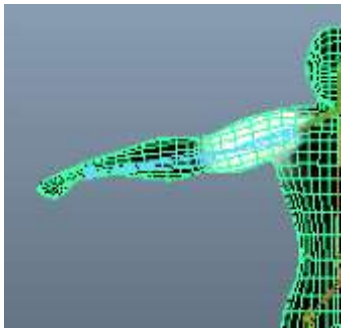
Left Hip



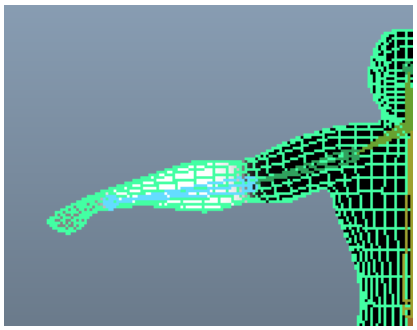
Left Knee



Neck



Left Shoulder



Left Elbow

The skeleton is animated correctly, however the skin is not moving properly with the skeleton. Since the skin weights seem to be correct, and the manual rotations are correct, we do not know the exact reason behind this discrepancy.

Results

Overall, our project turned out to be a success. We successfully completed our goal of translating the movements of a person into the movements of a graphical body. As

shown in the screenshots below, the Kinect was able to very accurately detect specific pivot points on the body and, using that information, we mapped those points onto a basic graphical model.

The major area for improvement on this project is making the models look fully realistic. Although our joint skeleton system is perfectly accurate, we had a problem with the skinning of the actual skeleton as mentioned before.

If this project were to continue, our next step would be to map these control points onto different rigged models (both human and non-human). A main trouble that we were having was the rigging of the model's themselves. We decided not to use pre-rigged models because all of the pre-rigged models (on TurboSquid) had different control points that our Kinect output, so, when mapping corresponding joints, the joints on the rig that did not map to a Kinect point were difficult to deal with and it was more of a hassle than a help. Therefore, we decided to hand rig all of our models in Maya. In addition, if we were to continue and map the human skeletons to non-human models (ex. a horse), we would have to have a correct mapping of human joints to horse joints.

We really enjoyed this project and we thank you for all the help and suggestions for making it happen. We had a great semester and learned an incredible amount. THANK YOU!

- Evan and Chandani

Final Video: https://www.youtube.com/watch?v=H7Ow2yNyVAA