INDRAPRASTHA INSTITUTE *of* INFORMATION TECHNOLOGY **DELHI**

CSE 533/333 Computer Graphics, IIIT-Delhi

Kinect Driven 2D Mesh Animation

Sarthak Ahuja (2012088)

Under the guidance of Dr. Ojaswa Sharma

INTRODUCTION

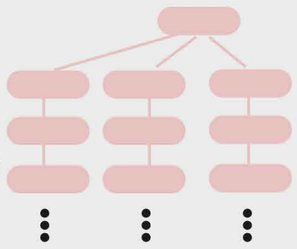
Kinect uses it's infrared sensor to calculate depth images of a scene. It further is capable of detecting human s in the frame and identify their joint coordinates. This project is aimed at animating a 2D character using these joint coordinates provided by kinect. Kinect 2.0 is used in this project to provide the skeleton which is mapped to our character in QT and rendered using OpenGL.

Current Constraints:

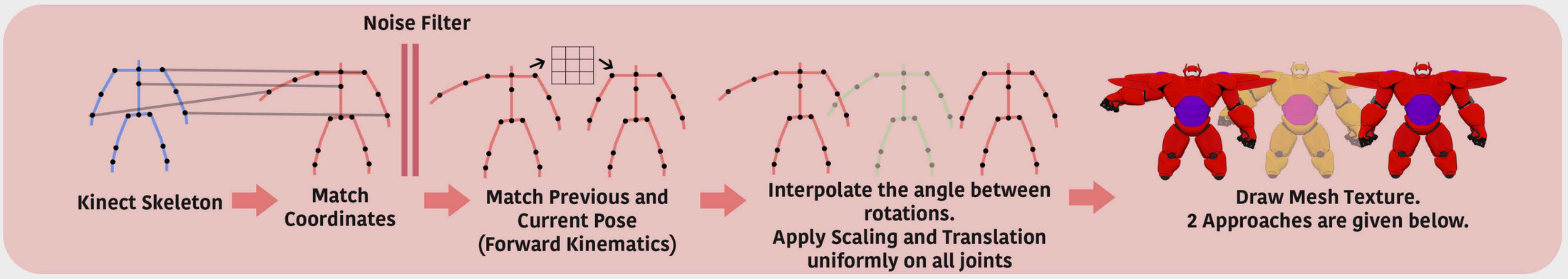
- Rigging between mesh skeleton and kinect joints is available.

UNDERLYING STRUCTURE

- The Character is setup as a **N-ary Tree**.
- The Nodes of the tree represent **joints**.
- These joints store **transformation matrices**.
- Each joint matrix affects it's children. (**Recursive**)
- The root node in our case is the **hip center**.
- In the current model we have **20** joints



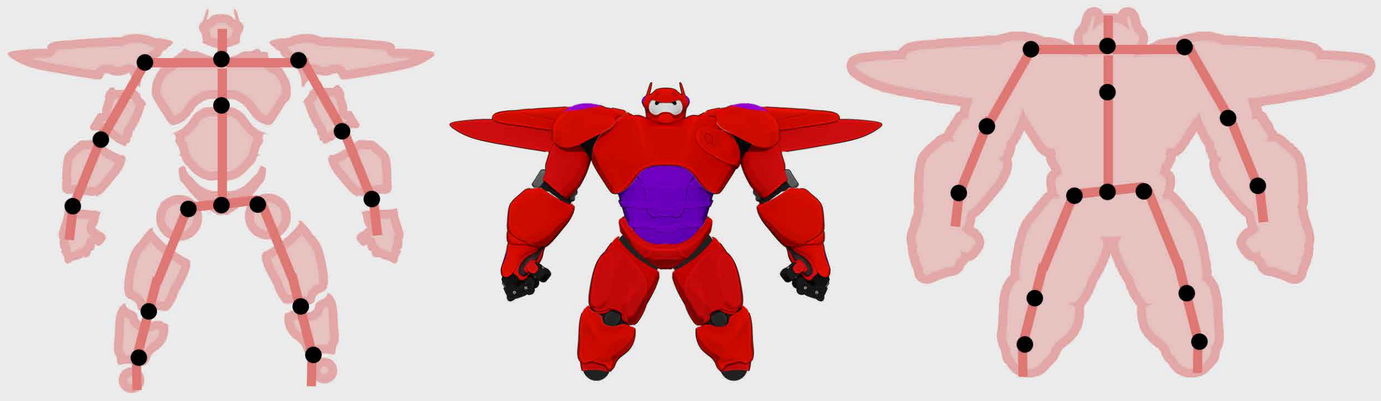
PIPELINE



Performing Mesh Transforms

Approach 1
Rigid Transforms

- **Separate Mesh** for every Body Part.
- Distance between two points in a mesh remains same throughout
- Gives **Puppet/Silhoutte** feel.
- Unable to clearly cover joints with mesh during animation.
- **Easy to implement**



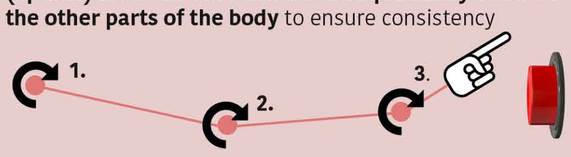
Approach 2
Non-Rigid Transforms

- **One Single Mesh** for entire skeleton.
- Transformation of mesh points as a weighted function of the neighbouring joints.
- Gives **Actual Character** feel.
- Able to cover joints with mesh during animation with reasonable clarity
- **Tricky to implement**

IMPLEMENTATION DETAILS

Forward Kinematics

- To function correctly we need to **begin rotations (update) from the root and move sequentially towards the other parts of the body** to ensure consistency



- Constraints
All body parts have certain constraints associated with their parents, so as to how much they can be rotated, moved or scaled. This is necessary to maintain structure and character.

Implementation Pipeline

- Data from Kinect is subscribed using Visual Studio and the joint coordinates are published as files in a buffer folder.
- The QT application picks up the files in sequence (simultaneously deleting them) and uses them as the input Kinect Skeleton Joint coordinates.

Classes and Code Structure

- Scene:
This is the main class containing the canvas and character.
- Character
This is the generic character class which contains the joint tree.
- Dataset
This is the class used to buffer in frames.
- Joint
This is the node of the created tree which forces children to implement the draw function.
- Transformation
This is the class used to create Transformation matrices.
- MainWindow
This is the main UI class.

- Baymax - Implementation of **Character**
- Thigh, Head ... - Implementation of **Joint**

FUTURE WORK

- Automatic Skeleton Generation and Mapping
As of now we are assuming we have a matched skeleton

- Extend the system to 3D
As of now we cannot deal with Scaling along z-axis.

CALCULATIONS AND OPTIMIZATION

- Noise Removal: A median filter is applied to remove noise in the joint coordinates received from kinect.
- Interpolation: Linear Interpolation is performed on the rotation angle.

EVALUATION

Besides live testing, evaluation done on the MSR skeleton dataset
http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/
16 activities, 30 subjects

REFERENCES

- Real-time physical modelling of character movements with Microsoft Kinect, **Shum and Ho**
- Motion Capture Applied To Sports, **Ashish Shingade and Archana Ghotkar**
- Animation of 3D Characters from Single Depth Camera, **M.May, F.Xu Y. Liu**
- Shape-aware MLS deformation for line handles, **Sharma, Ojaswa, and Ranjith Tharayil**