Arms animation based on human hierarchical tree model

Abstract. The paper presents a human model based on hierarchical tree structure, which is used for creating various 3D animations of the human body. Two animations are presented as examples. The model uses quaternions for representing rotations. It is implemented in C++ language. Two integrated software libraries are used for manipulating quaternions and writing standard C3D files which can be further viewed and processed in various programs such as Mokka.

Introduction

3D animation is a common way of visualizing motion data. There are many techniques to creating and controlling specific movements. In order to create a movement model, a way of representing angular position is needed. There are four main ways of representing angular position in 3D space: (1) fixed axes, (2) Euler angles, (3) rotation and angle and (4) quaternions [1]. They can all be used to rotate the 3D objects.

This paper presents a hierarchical human model and a method of animating two human arms in 3D. The human body is represented as the simplified model consisting of 15 vertices (only arms are animated). Each vertex corresponds to one of the human body joints. Quaternions are used for representing the angular positions of limbs in space. The animation takes into account the physical properties and limitations of the human body. As a result the animation is realistic.

For purposes of this paper, a computer program is implemented in C++. Two main libraries are utilized by the piece of software: the biomechanical toolkit (b-tk) [2] and Eigen [3]. B-tk is used to create and save an animation in C3D format [4]. Such a file can be opened in various programs meant for motion capture data viewing and processing. In our case animation is opened in Mokka [5] in order to visualise moving arms. The second library - Eigen - is used to manipulate quaternions describing the hierarchical model.

Representing rotations with quaternions

Quaternions are represented by four values \([s, x, y, z]\). They can be expressed as a pair \([s, v]\), where \(s\) stands for a scalar and \(v\) for a vector consisting of three coordinates \(x, y, z\) [1, 6]. They can be used for both representing rotations and for interpolating angular positions. They are also often used for combining various rotations into one transformation. All rotations can be represented by quaternions.

A point \((x,y,z)\) – i.e. a point in space -- can be represented by a quaternion whose scalar part is 0 and the vector part consists of \(x, y, z\) (equation 1) [1].

\[
v = [0, (x, y, z)].
\] (1)

Rotation of the point represented by \(v\) around the axis passing through the origin of the coordinate system can be performed by using quaternion multiplication as shown in equation 2 [1].

\[
v' = qvq^{-1},
\] (2)

where: \(q, v, v'\) stand for quaternion, point and rotated point correspondingly.

Compound rotations may be represented as a product of quaternions that represent consecutive rotations. A compound rotation of point \(v\) by two quaternions \(p\) and \(q\) is given by equation 3 [1].

\[
v'' = pvq^{-1}p^{-1} = p(qvq^{-1})p^{-1}
\] (3)

\[
= (pq)v(q^{-1}p^{-1}) = (pq)v(pq)^{-1}
\]

where \(v''\) stands for the point that was subject to compound rotation.

The inverse of a quaternion represents rotation of a point around the same axis and by the same angle but in the opposite direction (equation 4) [1].

\[
q^{-1}qvq^{-1} = \begin{cases} v & \text{if } \theta = 0; \\ \end{cases}
\] (4)

Rotation by an angle \(\theta\) around the axis determined by the vector \(v = [x, y, z]\) can be represented by a unit quaternion \(q\) given by the equation 5 [1].

\[
q = \cos(\theta/2), \sin(\theta/2)(x, y, z)
\] (5)

Rotation by an angle \(\theta\) is equivalent to a rotation by an angle \(-\theta\) around an axis whose orientation is the opposite. The quaternion \(q = [s, v]\) and its negation \(-q = [-s, -v]\) represent the same rotation [1].

As in the case of point rotation, in order to rotate a vector \(w\) a new quaternion has to be created \([0, w]\). It represents the vector being rotated. A second quaternion \(q\) describes the rotation. The rotated vector \(w'\) is given by the equation 6 [1].

\[
[w', 0] = [0, w]q^{-1}
\] (6)

Also with vectors compound rotations can be expressed by a product of several quaternions. For instance, two vector rotations described by two quaternions \(p\) and \(q\) can be compounded as shown in the equation 7 [1].

\[
[w', 0] = [p [0, w]p^{-1}]q^{-1} = ([qp][0, w][qp])^{-1}
\]

where: \(w''\) is the vector after the compound rotation.
As with points, the inverse of the quaternion represents the rotation of a vector around the same axis by the same angle but in the opposite direction. The combination of two rotations represented by $q$ and $q^{-1}$ produces the identity transformation as shown in equation (8) [1]:

$$q^{-1}(q[0, w]q^{-1}q) = [0, w]$$

### Human hierarchical tree model

An artificial human model is a hierarchical data structure. A human is represented with the use of a tree consisting of linked nodes. The top node is the root. Its coordinates are given in the global coordinate system. Coordinates of other nodes are defined in relation to the root. Each node has a parent (except the root) and at least one child node (exceptions are the lowest nodes called leaves). Each node corresponds to a joint and a specific part of the body that starts in the parent node (a node higher in the hierarchy).

In Fig. 1 a simplified human silhouette is presented. Joints/nodes are marked as red dots. Figure 2 shows the silhouette in a tree form. Nodes contain the names of the joints they correspond to. An arbitrary assumption is made as to which nodes represent the left and right side of the body.

![Tree root](image)

**Fig. 1. A simplified human silhouette**

![Human hierarchical tree model](image)

**Fig. 2. Human hierarchical tree model**

The back of the human silhouette is the root of the hierarchical tree. It divides the model into two parts corresponding to the bottom and top part of the body. The root has three child nodes: left hip, right hip and neck. Each of the two hip nodes has one child a knee, which in turn also has one child an ankle, which are leaves of the tree. The neck node is the start of the top part of the body. It has three children: left shoulder, right shoulder and the head. Each shoulder has one child – the elbow which in turn also has one child – the wrist. The wrists and the head are leaves of the tree.

The model imposes restrictions on the relative positions of objects it describes. When a specific joint is in motion, other related joints are also moved in a manner defined by the hierarchy. This means that all nodes located lower in the hierarchy will be moved according to a transformation defined for their parent. In other words, moving a joint is equivalent to moving a body part defined as a sub-tree consisting of all nodes (joints) taking part in this transformation.

The hierarchical model is implemented as a collection of classes in object oriented C++ language [7]. First is the Node class whose fragment is shown in figure 3.

```cpp
class Node {
    Node(string jointName);
    btk::Point::Pointer mokkaPoint;
    Vector3d v;
    Transform3d initialTransformation,
    articulationOfTransformation;
    std::vector<Node*> children;
    string partOfBody;
    Node* createNewNode(int numberOfChildren,
    string jointName);
    void update(Transform3d parentTransform,
    int frameNumber);
}
```

**Fig. 3. Selected fragment of Node class’ implementation**

The class consists of: (1) a constructor whose task is to prepare an object that represents joint named by the constructor’s parameter, (2) a BTK library’s point data structure (its value is written to C3D file), (3) current coordinates that represent the joint’s current position in space, (4) a joint’s initial transformation and its articulation, (5) a list of the node’s children (nodes that are lower in the hierarchy), (6) a joint name (naming joints makes model manipulation easier), (7) two methods: createNewNode() and update() which will be described in more detail below. The former creates a node based on two pieces of information: the number of children and a joint’s name. The latter method using two parameters (current node transformation and the frame number) computes a new node’s position and also updates the positions of all its children as shown if figure 4. As one can see that the positions are updated recursively.

```cpp
for(int i = 0; i < children.size(); i++){
    children[i]->update(newTransform,
    frameNumber);
}
```

**Fig. 4. Updating positions of node’s children**

A new node position is obtained by computing a product of matrices representing: the current node transformation, the initial node transformation and its articulation. The result is saved in the mokkaPoint field. The node’s updated position is needed not only for displaying purposes but also for com-puting the updated positions of all of the node’s children.

Second class: HierarchicalModel is responsible for: (1) constructing the hierarchical model using node objects
and (2) creating animations using the model. Class’
definition is shown in figure 5.

```cpp
class HierarchicalModel
{
    HierarchicalModel();
    Node *back, *leftHip,...
    void createHumanModel();
};
```

Fig. 5. Class' structure

The class’s structure is simple – it consists of the
constructor, pointers to 15 nodes that make up the model
and the createHumanModel() method that builds the
model and creates animations. Building the model consists
of creating 15 node objects, defining relations between
them and saving them in the pointer fields. The names are
added for the more intuitive use.

The createHumanModel() method contains all the data
necessary for building a model of a human silhouette and
simulates its movement.

Arms animation based on hierarchical human model

With the use of the hierarchical model two example
animations are created for the purpose of this paper.

Creating animation begins with setting up the model. It
is constructed in the following manner: (1) the node objects
are created for each joint represented by the model, (2) the
nodes are organized into a hierarchical data structure
whose root is the node representing the silhouette’s back.
An example of how the hierarchy is built is given in figure 6.
(3) Each node is assigned its initial transformation (this way
a human silhouette in its initial position is created), a new
Transform3D object represents the initial transformation,
(4) each node is assigned an object (saved in the
mokkaPoint field), which is in turn associated with a point
trajectory saved in C3D file, (5) When all the nodes are
setup, their positions are updated by a call to the update()
method.

```cpp
back->children.push_back(leftHip);
back->children.push_back(rightHip);
back->children.push_back(neckHip);
```

Fig. 6. Assigning 3 child nodes to the root (the "back" node)

The model is positioned in such a way that the “back
node is placed in the origin of the coordinate system, and
in its neutral position both arms are straight (fig. 7).

Once the model is set up, an animation can be created.
As stated above, two animations are created. The duration of
both animations is set to 5 seconds. The total number of
frames is 125. Each frame contains coordinates of all 15
points that make up the hierarchical model. The first depicts
a person who is moving his/her arm (as if he/she were lifting
weights). In order to describe the initial and final positions of
both wrists, the quaternions are used. Two quaternions
represent the initial and final position of right hand and two
quaternions represent the initial and final position of left
hand. The initial position of the left hand (qLeft in figure 8)
and the final position of the right hand (qRight) is the same
as their position in the initial setup of the model. This
orientation is represented by the identity quaternion (a zero
angle rotation). The left hand in its final position (q1Left)
and the right hand in its initial position (q0Right) are rotated
by 135 degrees around the X axis. Appropriate quaternion
is generated using AngleAxis class [3].

The animation created by iterating through all the
frames to be created. For each frame the joint’s relative
position p in the range of [0;1] is computed. Then the
quaternions describing the intermediate (between initial and
final) positions are obtained using the SLERP interpolation
method [8,9]. New articulation is assigned to the left and
right elbow nodes and the nodes’ positions are updated.

```cpp
for (int i =1; i <frameCount; i++)
{
    //compute relative position p in the range of
    //compute new position of the right arm
    qRight = qRight.slerp(p, q1Right);
    // compute new position of the left arm
    qLeft = qLeft.slerp(p, q1left);
    rightElbow->articulationOfTransformation =
    qRight; //new articulation of right elbow
    elbowleftElbow->articulationOfTransformation =
    qLeft; //new articulation of left elbow
    //recursively update positions of the nodes
    back->update(ident, k);
}
```

Fig. 8. Arm animation

As the animation progresses, the left hand bends while the
right straightens and then the situation reverses. The
bending angle varies from 0 (straight arm) to 135 degrees.
Two joints are animated: the wrist and the elbow. Selected
animation frames are presented in fig. 9.

The second animation is created similarly. It depicts
straight arms being lifted. The difference is that the
positions of both wrists and elbows change in this animation.
Initial and final positions of the animated nodes are given by
four quaternions, two for each arm. The initial rotation of
both arms is identical and set to -90 degrees around the X
axis. The final position of the right arm is -90 degrees and
+90 degrees for the left arm. They are both rotated around
the Y axis. Appropriate quaternions were created again with
the AngleAxis class.

Selected frames from the animation are presented in fig. 10.
Conclusions

This paper presents a hierarchical human model which is a base for creating animations of selected parts of the human body. The animations are saved to C3D files which is one of the standard formats. Quaternions are used for describing limb movements. The advantage that use of quaternions offers is that it eliminates side effects such as gimbal lock [1]. The animation is smooth and resembles actual human movement.

The hierarchical model is a tool that can be used for creating animation of the whole human body and also for comparison with real motion capture data.

REFERENCES


Authors:
dr inż. Maria Skubiewska-Paszkowska, Politechnika Lubelska, Instytut Informatyki, ul. Nadbystrzycka 36B, 20-618 Lublin, Email: maria.paszkowska@pollub.pl,
dr inż. Jakub Smolka, Politechnika Lubelska, Instytut Informatyki, ul. Nadbystrzycka 36B, 20-618 Lublin, Email: jakub.smolka@pollub.pl,
Uczestnicy projektu Kwalifikacje dla rynku pracy - Politechnika Lubelska przyjazna dla pracodawcy współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego.