KicKinect - Torwandschiessen

Marcel Bruckner, Kevin Bein, Jonas Schulz, and Chandramohan Sudar

Technical University of Munich

KicKinect

KicKinect is the implementation of the game *Torwandschießen* where the player tries to kick a football through two openings in a wall. One hole is aligned at the bottom left and the other hole is at the top right. The player tries to kick the Ball through either of the two openings and collect points (1 point for the bottom and 3 points for the top hole). Both holes are just slightly bigger than the diameter of the ball which makes it difficult to hit. When either of the two holes is hit by the ball, the corresponding points are accredited.

Figure 1: Welcome screen of KicKinect

Linear Blend Skinning

Linear Blend Skinning (*LBS*) is used to deform a mesh by utilizing its skeletal structures. This is especially interesting to render human body or animal movements. *LBS* is a very efficient algorithm due to its linear character and as a result can be used for real-time renderings. To calculate the per vertex translation and rotation the kinematic chain of the bone structure is used.

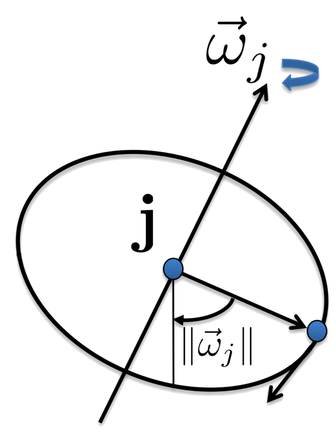

Figure 2: The bone orientation vector

For this project, *LBS* in its basic form is implemented. The translation vector of the bone is defined as the joint position j, $\vec{\omega}_j$ is the scaled axis of the rotation and $||\vec{\omega}_j||$ is the angle of rotation. Using the Rodrigues formula the rotation matrix R based on the bone orientation $\vec{\omega}_j$ is calculated as:

$$\mathrm{R} = e^{\hat{\vec{\omega}}} = \mathrm{I} + \hat{\vec{\omega}} \sin ||\vec{\omega}_j|| + \hat{\vec{\omega}}^2 (1 - \cos ||\vec{\omega}_j||) \qquad (1)$$

The rotation matrix R forms together with the translation vector j the overall per vertex transformation matrix G for the bone:

$$G(\vec{\omega}, j) = \begin{pmatrix} [e^{\hat{\vec{\omega}}}]_{3\times 3} & j_{3\times 1} \\ 0_{1\times 3} & 1] \end{pmatrix} \qquad (2)$$

Due to the natural chaining of the bones along the spine and extremities the calculation of adjacent joints corresponds to a chaining of transformation matrices:

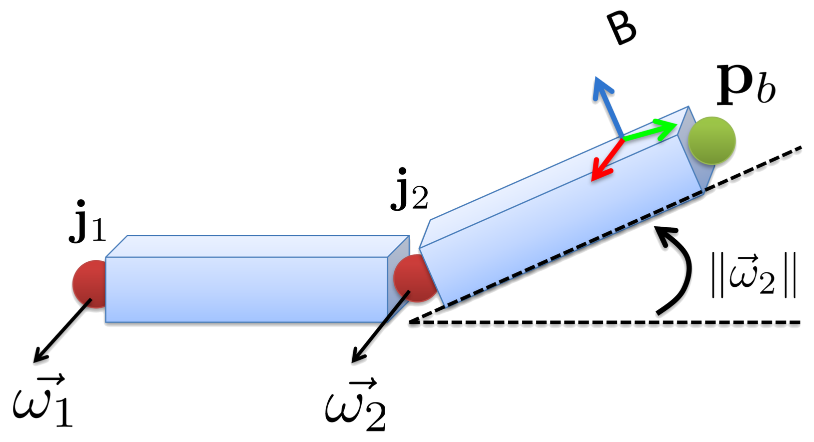

Figure 3: The kinematic chain along adjacent joints

The final transformation matrix G of the bones along the kinematic chain is calculated as a concatenation of per bone matrices $G(\vec{\omega}_j, j)$:

$$G(\vec{\omega}_1, \vec{\omega}_2, \ldots, \vec{\omega}_k, j_1, j_2, \ldots, j_k) = G(\vec{\omega}_1, j_1) \cdot G(\vec{\omega}_1, j_1) \cdot \ldots \cdot G(\vec{\omega}_k, j_k) \qquad (3)$$

To calculate the mesh deformation based on the kinematic chain transformations G the rest pose vertices $\bar{t}_i$ and the per vertex assigned bone weights $w_{k,i}$ are used. The final transformed vertex position $\bar{t}'_i$ is calculated by using:

$$\bar{t}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, J) \bar{t}_i \qquad (4)$$

Overview

A Microsoft Kinect Sensor is used to track the player and his movements. From this input data and the kinect SDK, a body skeleton is calculated and extracted (TODO: figure 2). By using *Linear Blend Skinning* this skeleton is mapped onto the 3D Model of a football player in Unity. For this game, it is especially important to detect movements of the lower body: the player's hips, knees, legs and feet. With the movement mapped to the model, the only thing left is to construct hitboxes around the players feet.

The scene consists of the Torwand and a ball, both which are hittable. Since the implementation of the collision detection and physics calculation are not directly related to this project, we are using Unity's build-in features. Still, the model parameters, the position of the ball relative to the player, the hitbox sizes and shapes as well as the velocity calculation for when the player hits the ball need to be optimized and tuned manually. With this setup, the only thing left is for the player to hit the ball with an appropriate angle and accelerate it towards the Torwand.

Figure 2: Shooting scene

Challenges

Ball velocity

Calculating the velocity with which the ball is accelerated towards the Torwand must be tuned manually. The following aspects account for a realistic value:

1. The distance of the player to the Torwand
2. The overall scaling of the scene
3. The size of the ball in relation to the player
4. The amount of hitboxes around the players feet as well as their shape
5. The angle of the hit and the rapidity of the body movement when swinging the foot
6. The mass of the Ball and the foot

The first four aspects can be tuned directly in Unity's physics and modle framework by modifying the parameters of the models and rigid bodies. For the fifth and sixth aspect, applying the formula of an elastic collision yielded very good results:

$$v = (2 \cdot m_{\text{ball}} \cdot v_{\text{ball}} + (m_{\text{foot}} - m_{\text{ball}}) \cdot v_{\text{foot}}) / (m_{\text{ball}} + m_{\text{foot}})$$

Synchronization

The biggest challenge comes from synchronizing the Kinect SDK with Unity. The Kinect is recording an input stream with a depth map and through the SDK, it provides the joints of the body (25 joints in total) and their rotations. Even though, ultimately, thes values are correct, the joints and rotations do not follow the convention of literature on this topic. MISSING EXPLANATION