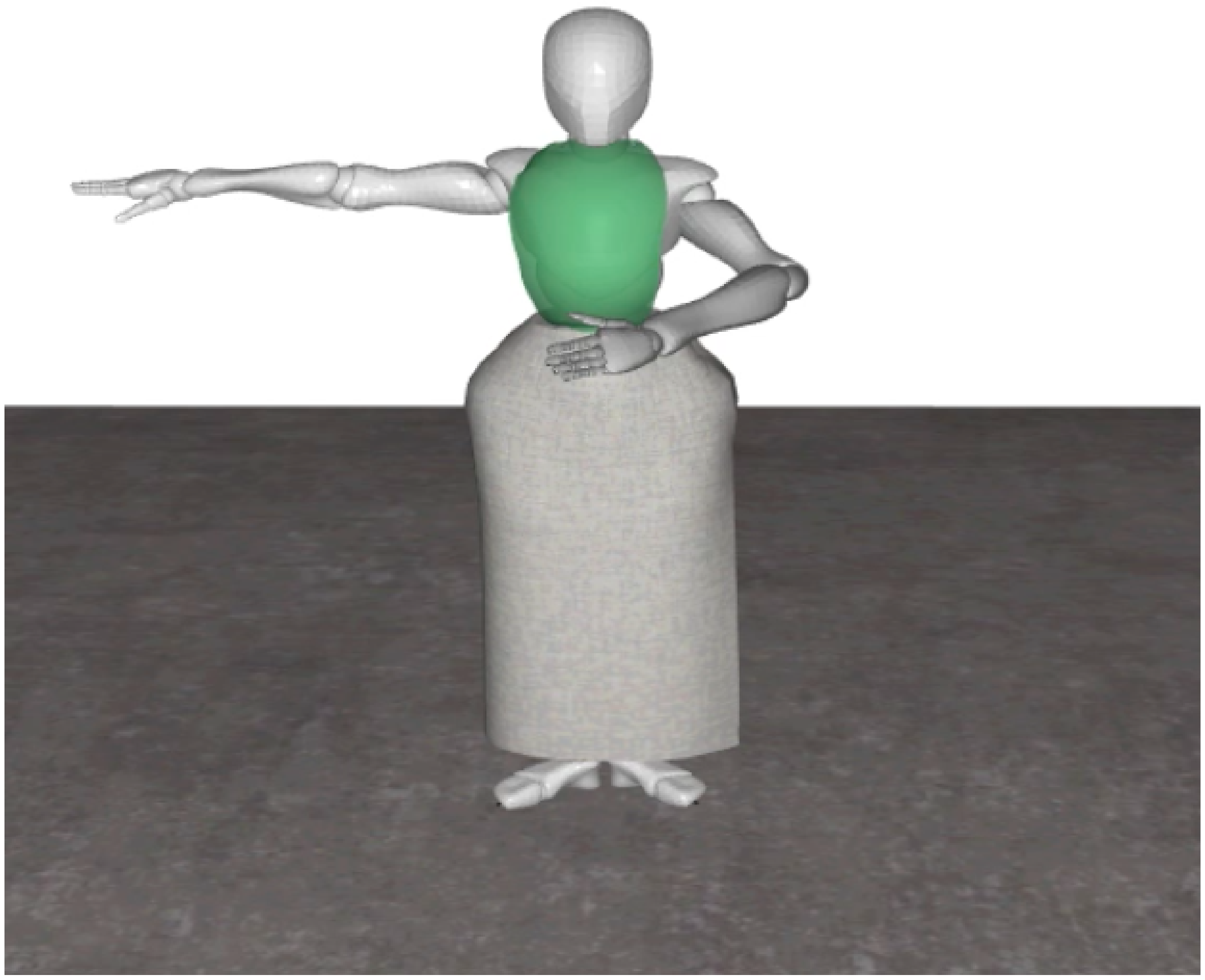
INF585 Project : body and cloth animation, a choreography

Capucine Leroux

March 2021



Contents

1 Introduction

The animation scene I decided to create represents a woman dancing around with a piece of cloth on, with the movement of the cloth adapting to the body movements. There are two main subjects to tackle in this project. First, we want to animate the body, this means building an interactive interface to create the poses of the skeleton, then interpolate these key poses and add some skinning for the mesh around. And second, we want to add some cloths on the body, which requires a proper initialization of the cloth shape, size and position on the body, and handle collisions. The final animation is rendered in real time.

2 Body animation

2.1 Interactive interface

The first step was to create an interactive interface to move the skeleton easily with the user mouse in order to have key poses for the choreography animation. The skeleton works with a hierarchical representation (joints are linked to their parent joints to create bones), which means that each joint is represented by a translation and a rotation in the parent local coordinate system.

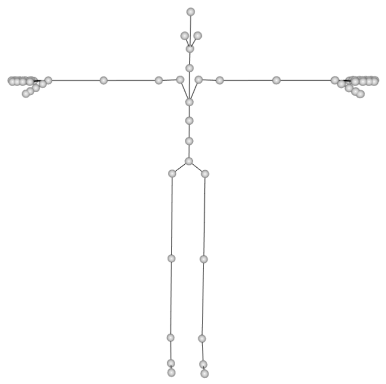


Figure 1: Skeleton hierarchy

We have several choices to control the movement of a joint. Here, I chose forward kinematics for simplicity, which means the previous bones in the hierarchy won't move, but only the joint we select. Since we don't need to make very difficult poses and we don't really mind if the feet are not always exactly on the ground here, it was simpler to implement.

In the interface I made, we need to press the Shift key to select a joint, then when a joint is selected, it can be moved in the camera plane with one constraint : the bone between the current joint and its parent is solid so the distance between the joints must stay constant. Therefore, the current joint can only move on the sphere of center p_{parent} and radius $||p_{joint} - p_{parent}||$. To compute the new position of the joint, I projected the position of the user mouse onto this sphere.

Then comes the question of what transformation to apply to the skeleton when moving the joint. If we only translate the current joint, it creates a problem afterward for the skinning of the mesh around it because it won't rotate the vertices properly. Similarly, if we add a rotation to the current joint, it will still create an issue for the skinning around the parent joint.

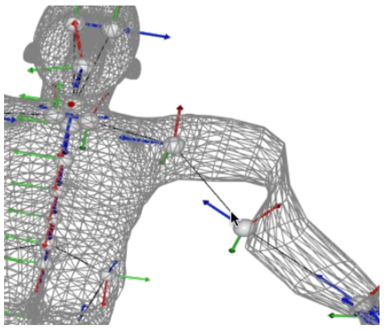


Figure 2: Skinning problem when transformating the current joint

That is why I applied a rotation to the parent joint instead. It requires to be careful about the coordinate system we are using. We rotate the parent in the local coordinate system doing :

$$R_{parent,local} = (R^{-1}_{grand-parent,global} \times R(\theta, n) \times R_{grand-parent,global})R_{parent,local}$$

where $\theta = arccos(\frac{(p-p_{parent}).(p_{new}-p_{parent})}{||(p-p_{parent})||||(p_{new}-p_{parent})||})$, $n = \frac{(p-p_{parent})\wedge(p_{new}-p_{parent})}{||(p-p_{parent})\wedge(p_{new}-p_{parent})||}$.

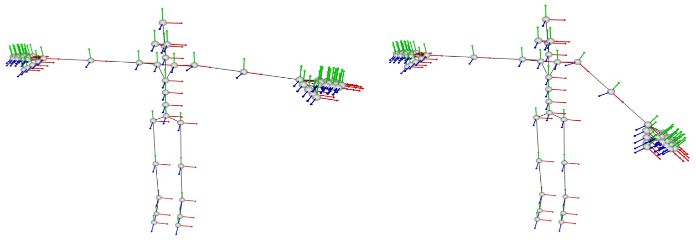


Figure 3: Rotation on the parent joint

This solution has still several limitations. First, we need to be careful about the joint root, which neither has a parent nor a grand-parent joint. For this joint, we apply a simple translation to it. Second, in this solution, moving a joint has a consequence on all of its descent, which is not necessarily an issue in our case. Also, the positions are sometimes hard to control since the translation is made regarding the view plane. It can be hard to give the right orientation to the joint from a single view. Last, it means that if a parent has several children, moving one children moves all the other children. Here, the only body members that are concerned by this issue are the fingers and the two hips and shoulders. As we do not really need to move the fingers here it is not a problem, as for the others, I tried to prefer moving arms and legs before hips and shoulders to avoid any conflict. An future improvement could be to separate the parent joint into several pseudo-entities according to which bone is in action.

2.2 Skin deformation

Now that the skeleton moves interactively, we need the body mesh to follow this movement. For that, I used linear blend skinning. As the body doesn't get very small angles between two bones, we don't necessarily need dual quaternion skinning for this purpose, especially with the mesh I used which has spheres on most articulations, but it could be an improvement.

Linear blend skinning :

$$p = (\sum_{k=0}^{N-1} \alpha_k T_k (T_k^0)^{-1}) p^0$$

where α_k is the weight of joint k for this vertex, T_k is the current transformation of joint k, T_k^0 the rest pose transformation of joint k, and p^0 the rest pose position of the current vertex. The rig weights are already known at this point (See paragraph 2.4).

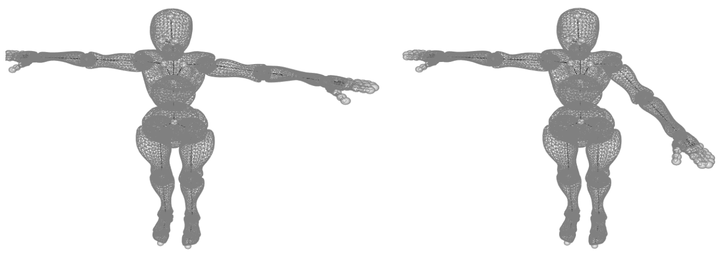


Figure 4: Skinning

2.3 Movement animation loop

Now we have all the elements to create an animation loop to make the body dance. The key poses are pre-saved in a text file. The interactive interface enables us to add a pose to the file by pressing 'S' and save the final files by pressing 'F'. The final files consists of three texts, one for the joint translations, one for the joint rotations and one for the different times of the different poses.
Once we have all the different key poses, we can visualize the animation loop by interpolating those key poses through time, the skinning deformation is applied in real time during the simulation.

Both the translations and the rotations are linearly interpolated.

For the translations, at the time t between two poses 1 and 2:

$$T = (1-t)T_1 + tT_2$$

For the rotations, we use the quaternion representation to apply LERP interpolation:

$$q = \frac{(1-t)q_1 + tq_2}{||(1-t)q_1 + tq_2||}$$

One limitation to this interpolation is that the movement of the body does not take physical properties into account at all. That makes it a little difficult to render a realistic movement when the body is jumping for example, since the evolution of the speed of a jump is never perfectly linear in reality.

2.4 Format pre-processing

As we mentioned earlier, in order to animate the body, we need several pieces of information, which are the skeleton rest pose data and hierarchy, the mesh rest pose data and the weights of each joint for each vertex. For this, we need a particular type of format made for skinning. In this project, I used an open source collada (.dae) file [1]. It required a pre-processing step in order to extract all of the needed information. A particularity of this precise file is that the body is separated into two different meshes and skeletons that I had to put in common.

3 Cloth animation

Now that we have animated the body, we want to put clothes on it. Here we model a cloth by a rectangular grid of particles. Then the animation is physically-based. A gravity force, a drag force and spring forces with neighbor particles are simulated on each particles, then the velocity and the position are integrated with a semi-implicit euler method. This model gives some elasticity to the cloth. Then the cloth must keep out of the body mesh in order to look real.

3.1 Constraints

To handle the collisions with the body in real time, I approximated the body shape with simple geometric shapes such as spheres and rounded cylinders. Searching for collisions between each particle and each face of the mesh took too much time to be rendered in real time. The same goes for the self-collisions which were too costly to compute.

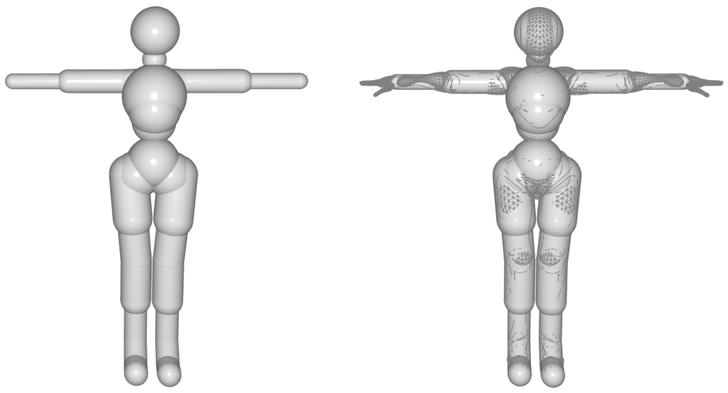


Figure 5: Proxies

The collision between a particle p and a sphere goes as follows :

If $||p - center_{sphere}|| < radius_{sphere}$: *(collision)*

$n = \frac{p - center_{sphere}}{||p - center_{sphere}||}$
$p = center_{sphere} + radius_{sphere} n$
$v = v - (v.n)n$

The collision between a particle p and a cylinder of extremities p_1 and p_2 goes as follows :

If $projection_{(p_1,p_2)}(p) \in [p_1, p_2]$ and $||p - projection(p)|| < radius_{cylinder}$: *(collision)*

$n = \frac{p - projection(p)}{||p - projection(p)||}$
$p = projection(p) + radius_{cylinder} n$
$v = v - (v.n)n$

The rounded cylinder is a cylinder with two semi-spheres at the extremities, to handle the extremities as well.

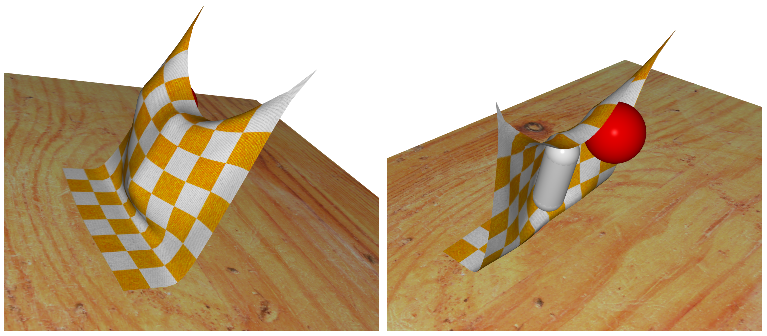


Figure 6: Cloth collision with a sphere and a rounded cylinder

We can also add the collision with the ground to make sure the cloth doesn't go below a certain height. A limitation of these algorithms is that if the grid is too large compared to the proxy obstacles, a collision can be missed and result in serious issues and divergence.

Apart from the external constraints, the cloth has to stay in a certain shape on the body. For instance, the skirt needs to remain around the waist. For that, we add internal constraints on the borders of the cloth for it to keep in a circle around the waist joint. We can add other details such as to keep the skirt closed until a certain height. Those constraints need to be manually computed to fit the real body size, in order to avoid conflicts between internal and external constraints. Even so, it is sometimes hard to make sure the constraints keep moving with the body. For example, when closing the skirt, in order to make sure the seam stays in place, I gave a fixed position to the vertices. However, when the hip moves too much, the vertices on the seam don't move appropriately.

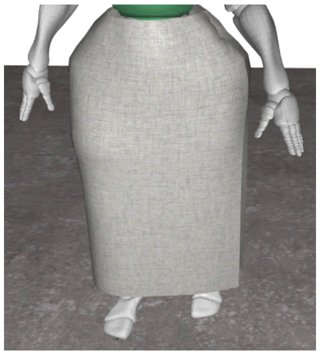


Figure 7: Internal constraints for a skirt

3.2 Mesh initialization

One important factor when putting a cloth into the simulation is to initialize properly its initial position, since the future positions of the particles will only be determined by their initial value and the physical simulation. For example, it can be problematic to initialize the cloth where an arm will be at the beginning, because the collisions will be handled difficultly and it can result in a simulation divergence.
One solution for that is to pre-compute the initial positions of the particles, by taking a simple initial shape for the cloth, waiting for it to stabilize on a still body, and save the stabilized positions.

4 Conclusion

In conclusion, the project handles body interactive animation and cloth animation with constraints. The simulation works in real time with very few pre-computations other than the skeleton key poses. In order to look more realistic, we could use inverse kinematics instead of forward kinematics to have better control over the extremities of the skeleton hierarchy, or duplicate some joints to handle multiple children separately. For the cloth, we could complicate the proxies to have more realistic collisions, and use more complex shapes than rectangles for the cloth to get more realistic cloths.

References

[1] *Collada file reference*
https://www.mixamo.com/#/?page=1&type=Character