1 Introduction

“Pirates of the Caribbean 2: Dead Man’s Chest” features Davy Jones, an all CG character, whose most outstanding feature is his beard, designed to look like an octopus with dozens of tentacles. The beard presented multiple challenges for animation, simulation, and deformation. This sketch will elaborate on techniques developed for the rigid simulation of Davy Jones’ beard. To make this character believable, it was critical that the tentacle behaved like that of an octopus, but still presented the dynamic motion of the character’s performance. ILM utilized Stanford’s PhysBAM simulation system as a base for the articulated rigid body simulation that drives the performance of the tentacles. Along with Stanford’s solver, ILM created animation controllers that allowed the artists to direct the performance of the simulation. By incorporating the Stanford solver into our proprietary animation package, Zeno, we made it possible for a team of artists to quickly produce the animation for the 200+ shots required by production.

2 Dynamic Tentacles

An articulated rigid body dynamics engine was utilized to achieve the desired look. Each tentacle was built as a chain of rigid bodies, and articulated point joints served as a connection between the rigid bodies. This simulation was performed independently of all other simulations, and the results were placed back on an animation rig that would eventually drive a separate flesh simulation.

Since Davy’s beard had 46 tentacles with a total of 492 rigid bodies and 446 point joints, a controller system was needed in order to make the simulation manageable for an artist. Each tentacle had a controller to define parameters to achieve the desired dynamic behavior, which was mainly influenced by the head motion and any colliding objects. Another controller, which served as a multiplier for all individual controllers, helped the artist to influence the behavior of the whole beard at once. To make the tentacles curl, the connecting point joints were motorized using a sine wave that was controlled by attributes like amplitude, frequency and time. Most dynamic parameters were set along the length of the tentacle. So, in order to automate the setting of these parameters, a 2D curve represented the length of the tentacle on the x-axis and the value of the parameter on the y-axis. Periodically some tentacles required key framed animation in order to manipulate objects in the scene. When specific interactions were required from the animation pipeline, the rigid bodies were set to follow the animation and used as collision geometry for the simulated tentacles.

The control for each joint on a tentacle was accomplished using a force-based targeting system. The targeting system calculated torques between rigid objects constrained by a joint. The goal of the targeting system was to minimize the difference between the joint’s target angles and current angles. During the progression of the simulation, the target angles for the joints were modified by the animation controller. For each joint, the targeting system calculated the difference between the target orientation and current orientation. The resulting difference produced an axis of rotation which defined a torque around which the connected rigid objects rotated. The final step was to apply the calculated torque back onto the connected rigid objects.

A real tentacle has numerous suction cups that allow the tentacle to momentarily grab onto surfaces and let go at any time. A functionality was required, termed “Stiction”, which would automatically create connecting springs between rigid bodies, to correctly mimic this momentary grab and release. The Stiction-spring interface was implemented through a set of area maps on the underside of the tentacle, defining regions on the tentacle where springs could be created. Properties of the Stiction interface defined distances at which springs could be created or destroyed, thus displaying the correct motion of the grab and release.
