Newtonian Physics Sandbox

Second Evaluation results

Evgeny Dedov

Goals

1. Make work system on three test scenarios: two spheres collision, Newton's cradle, billiard balls
2. Make possible import from blender-compatible 3D scenes, contain spheres.
3. Testing / profiling different HPX parallelization options in LGD

Milestones

I. PovRay visualization + user controllers (configure initial state)

The goal was to make a user-friendly interface, user-friendly in the source code, to create the initial state of the scene, color configuration, background, camera. To implement this, when adding an item to the system, we associate it with a name that represents a std::string object or an integer, for further access to the object.

Also, there are often cases where there is no need to give each object a name, but there are groups of objects that we would like to customize. Therefore, you can also add an object to the group and edit whole group

The geometric parameters of the object, as well as the physical density value, are set when the object is created. You can do any manipulations on the object, such as adding speed, turning an object, either before placing the object in the system, or after, using the object name, or by changing the group of this object.

Special attention should be paid to the customization of visualization, using the PovRay toolkit. We can optionally adjust the texture, color, transparency and other visual options available in PovRay.

Modification of the camera has several parameters. The first ones are based on the parameters in PovRay - the position of the camera in the coordinate system - location, the object the camera is aimed at - look_at, as well as the aspect ratio of the screen (4/3, 16/9, etc.). In addition, you can change the distance to the location, along the line connecting position and location.

II. Developing test scenarios: Newton's balls, billiard balls, 2 spheres moving on collision course

Illustrative examples of classical mechanics in several variations to use all the features of user-controllers and also import from Blender-compatible format, namely .OBJ

III. Initializer for Blender exported files.

The .obj format was chosen as the easiest for parsing. It is a set of vertices, vertex normals, and also a set of polygons, which is a set of vertex groups that form a polygon. In order to use imports from scenes of spheres and objects composed from spheres, a rather simple scheme was used: first, we read all the vertices related to spheres. After this stage, we have an empty graph consisting of vertices. Further, reading the polygons, links are established between the vertices, several connectivity components are gradually formed - the vertices are simply divided into sets by the principle - the two vertices are in the same connectivity component if and only if there is a path along the polygons between the vertices. After processing all polygons, each of the connectivity component represents a set of vertices that belongs to a particular sphere. In order to calculate the center of the sphere, we take the arithmetic average of the coordinates of the vertices. In order to find the radius, calculate the distance between the center and any vertex.

The HPX parallelization options research was postponed, because there were some problems with working capacity, and this stage had to be suspended