simulation

Notes

This project is implemented in Rust using the kiss3d library. I wasn't able to find a working rust install anywhere in the department file system (in theory, /course/cs1680/contrib/ should have one, but the permissions are wrong). A docker container which can run the project can be found here. I have also included a binary in the `bin/` directory. This binary runs on the department machines and will probably run on any reasonably recent version of linux.

Features

- Mesh - sphere intersection is implemented Due the way this is implemented, mesh sphere intersection will not work well in cases where the sphere is close to the size of a face on the mesh. For instance, intersection between the sphere and a single tet doesn't work well.
- Sliding friction between the floor and the mesh
- RK4 integration is implemented
- Node force calculations are parallelized (when this improves performance)

Videos and more details are described in the examples section.

Running the simulator

Most options should be pretty intuitive, and all options can be viewed with `./bin/simulation --help`. If the time step is too small to simulate in real time the window will lag and a warning will be printed in the terminal. A mesh file and a integration method must always be specified. Here are some examples of valid commands:

```
./bin/simulation meshes/sphere.mesh euler    # euler integration
./bin/simulation meshes/sphere.mesh midpoint # midpoint integration
./bin/simulation meshes/sphere.mesh rk4      # rk4 integration

# reduced incompressibility
./bin/simulation meshes/sphere.mesh --incompressibility 10.0 rk4

# reduced rigidity
./bin/simulation meshes/sphere.mesh --rigidity 10.0 rk4

# increased viscous rigidity
./bin/simulation meshes/sphere.mesh --viscous-rigidity 10.0 rk4

# increased viscous incompressibility
./bin/simulation meshes/sphere.mesh --viscous-incompressibility 10.0 rk4
```

```
# larger time step (explodes)
./bin/simulation meshes/ellipsoid.mesh --time-step 0.01 rk4

# no floor friction
./bin/simulation meshes/ellipsoid.mesh --floor-friction-coeff 0.0 rk4

# changing sphere
./bin/simulation meshes/ellipsoid.mesh --sphere-radius 2.0 --sphere-pos-x -2.0 rk4

# changing floor location
./bin/simulation meshes/ellipsoid.mesh --floor-pos -5.0 rk4
```

Examples

Videos can be found in the `outputs/` directory. Some of the command line arguments are specifically for producing videos at 30 fps. Here are the commands used to produce all of the videos and explanations of what they show:

```
# default settings ellipsoid
outputs/default_ellipsoid.mp4
./bin/simulation meshes/ellipsoid.mesh --hide --record-image-dir outputs/ \
  --frame-limit 360 --force-sim-fps 30 rk4
./scripts/make_video.sh outputs outputs/default_ellipsoid.mp4

# default settings sphere
outputs/default_sphere.mp4
./bin/simulation meshes/sphere.mesh --hide --record-image-dir outputs/ \
  --frame-limit 360 --force-sim-fps 30 rk4
./scripts/make_video.sh outputs outputs/default_sphere.mp4

# "bouncy" settings ellipsoid
outputs/bouncy_ellipsoid.mp4
./bin/simulation meshes/ellipsoid.mesh --hide --record-image-dir outputs/ \
  --frame-limit 360 --force-sim-fps 30 --rigidity 500.0 --viscous-rigidity 1.0 \
  rk4
./scripts/make_video.sh outputs outputs/bouncy_ellipsoid.mp4

# no friction ellipsoid
outputs/no_fric_ellipsoid.mp4
./bin/simulation meshes/ellipsoid.mesh --hide --record-image-dir outputs/ \
  --frame-limit 360 --force-sim-fps 30 --floor-friction-coeff 0.0 rk4
./scripts/make_video.sh outputs outputs/no_fric_ellipsoid.mp4

# default settings ellipsoid - stable with rk4
outputs/rk4_stable.mp4
./bin/simulation meshes/ellipsoid.mesh --hide --record-image-dir outputs/ \
```

```
  --frame-limit 360 --force-sim-fps 30 --time-step 0.0035 rk4
./scripts/make_video.sh outputs outputs/rk4_stable.mp4

# default settings ellipsoid - unstable with midpoint (explodes)
outputs/midpoint_unstable.mp4
./bin/simulation meshes/ellipsoid.mesh --hide --record-image-dir outputs/ \
  --frame-limit 360 --force-sim-fps 30 --time-step 0.0035 midpoint
./scripts/make_video.sh outputs outputs/midpoint_unstable.mp4
```

I found that parallelization improves performance for meshes with more than 300 vertices by up to 300% (the performance improvement is better the larger the mesh is). On all of the provided example meshes, threading reduced performance (synchronization/start up cost was too high). It takes a very large amount of time to simulate a complex mesh with enough steps for stability, so I haven't included any videos.