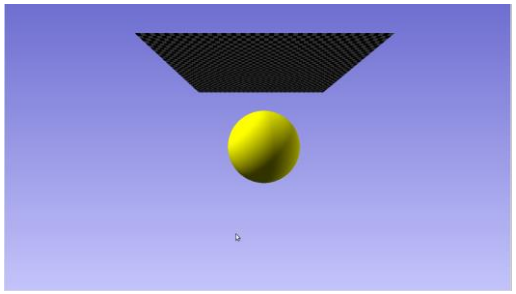
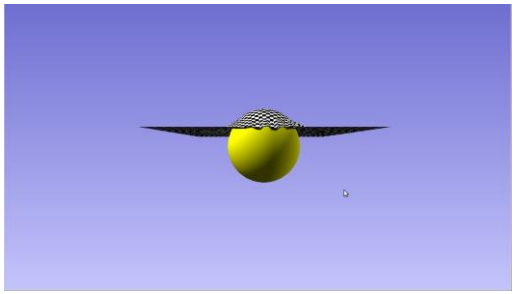
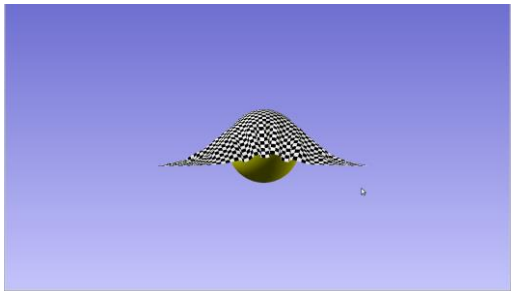
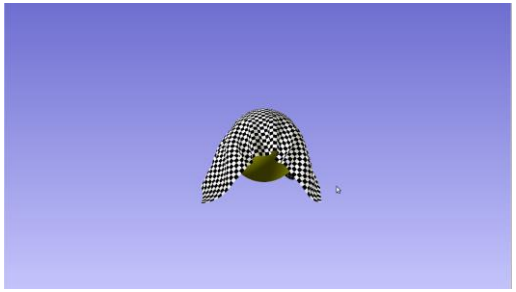
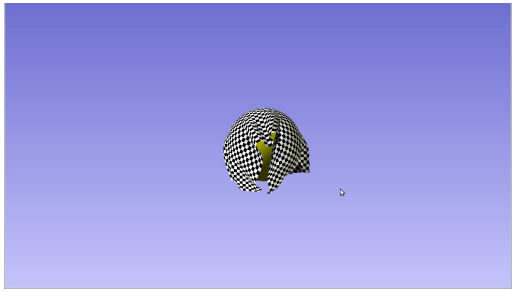
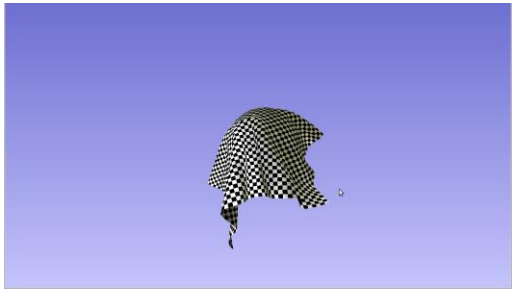
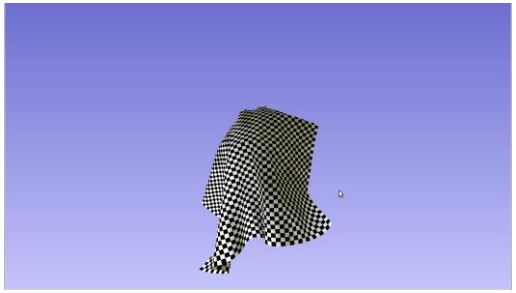
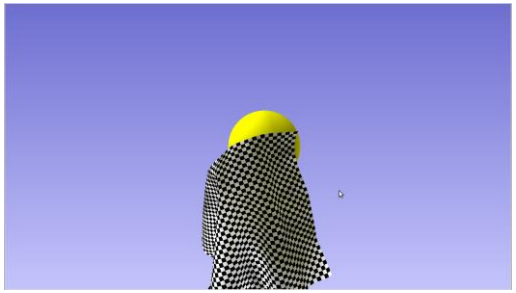
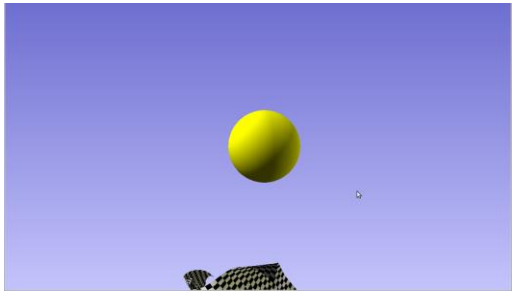
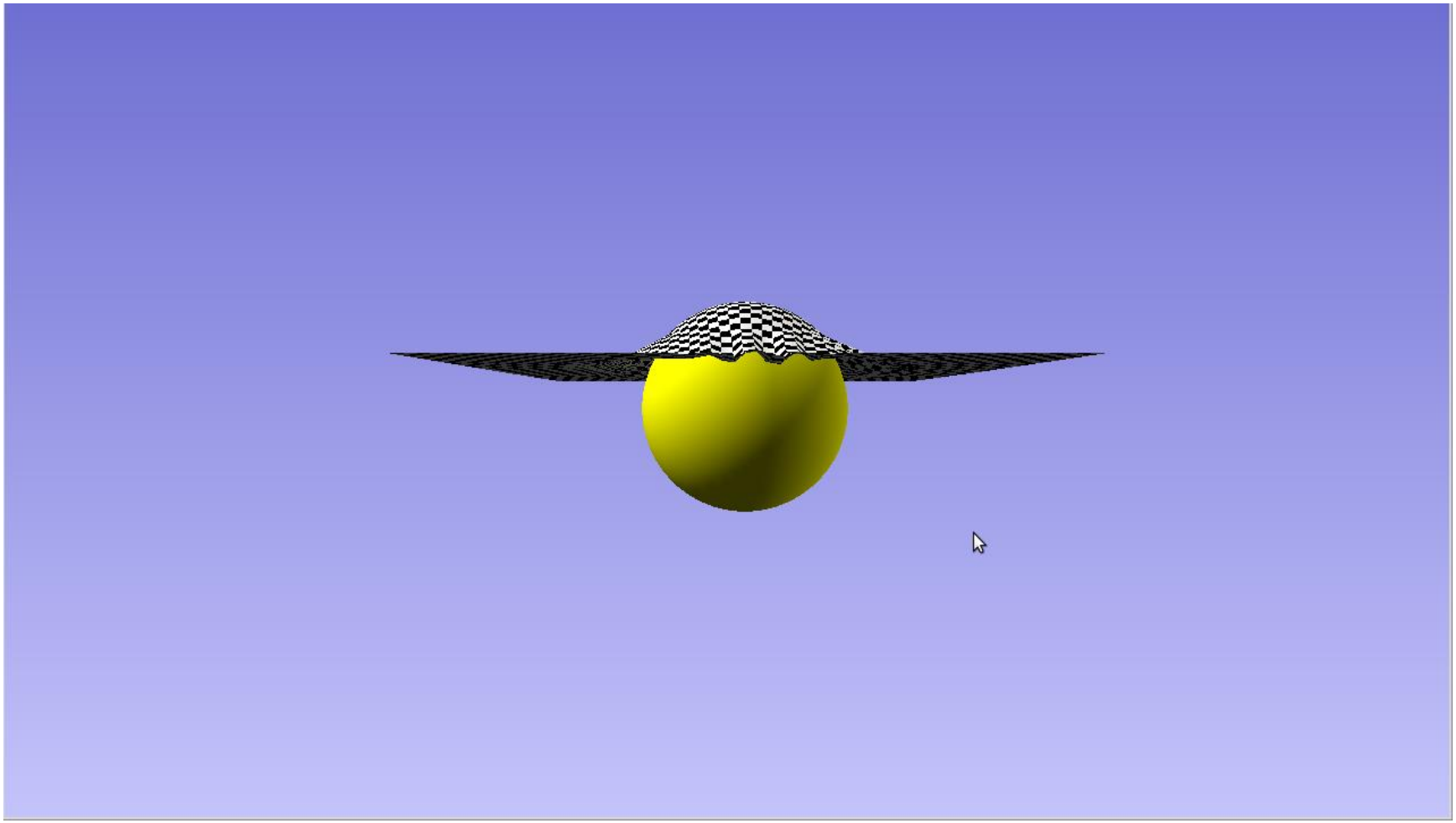
Collision Detection

CS-116B: Computer Graphics Algorithms
Spring 2018

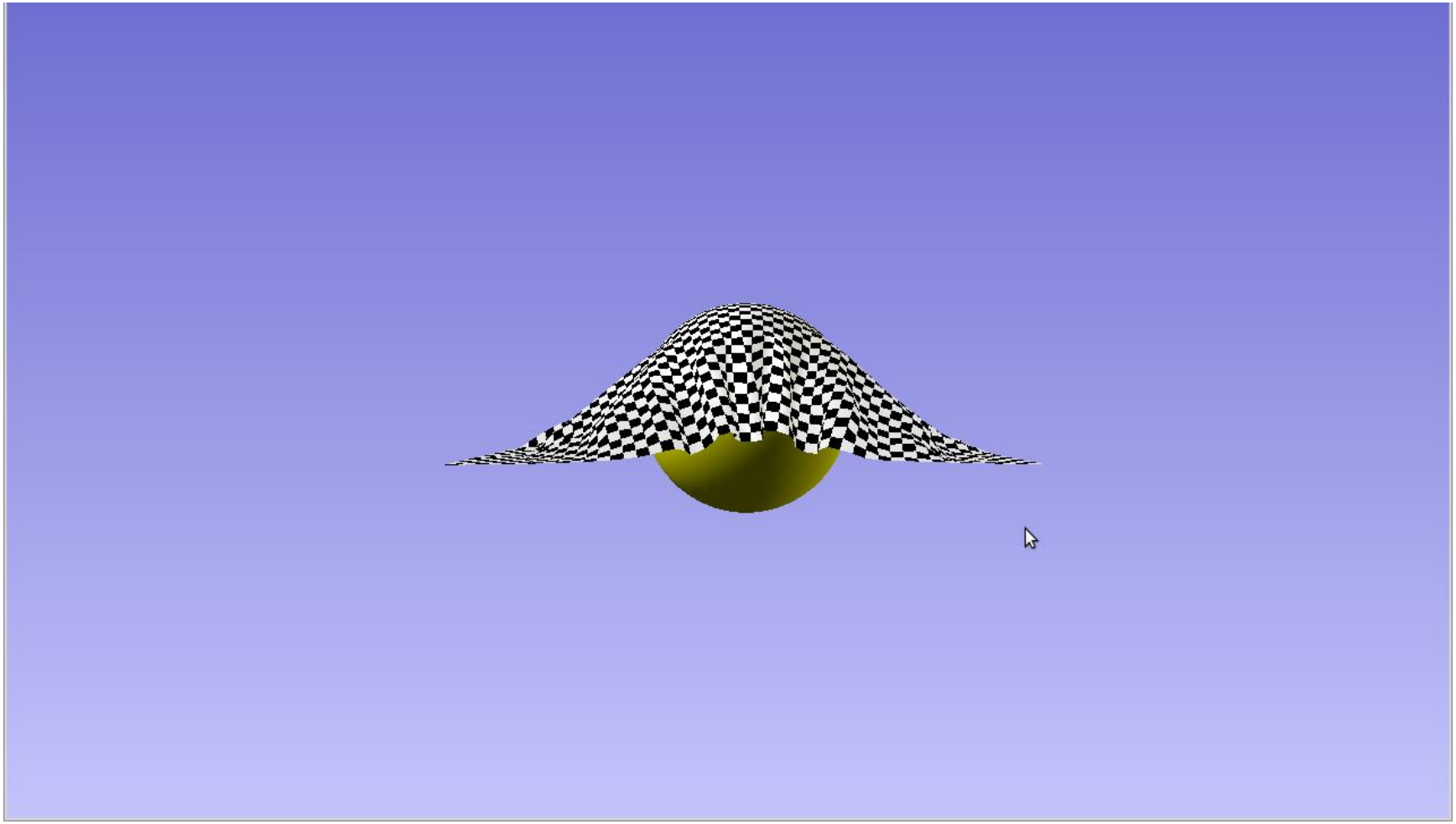
Cloth falling onto a stationary sphere

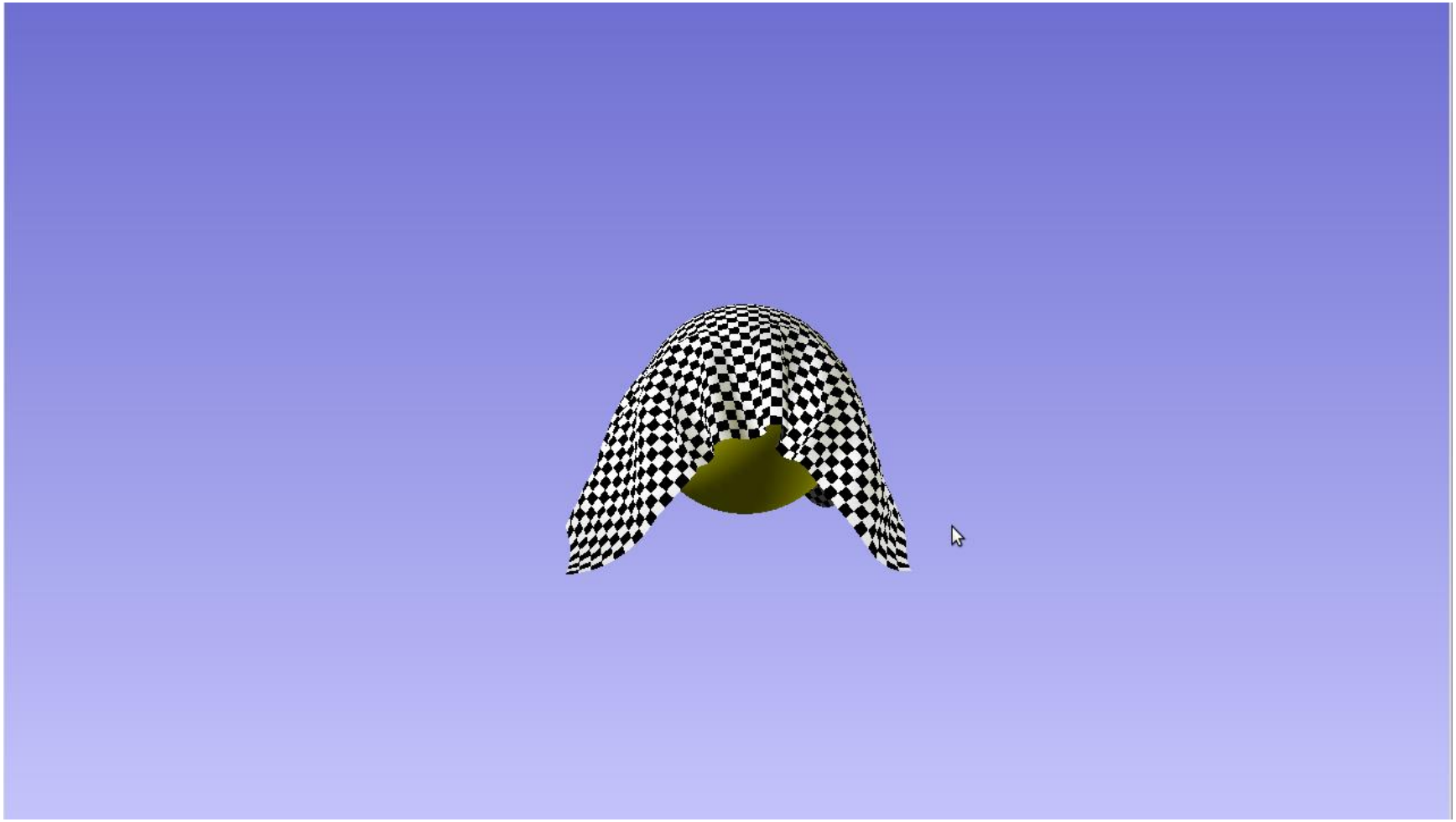
Closeup: Cloth falling onto a stationary sphere



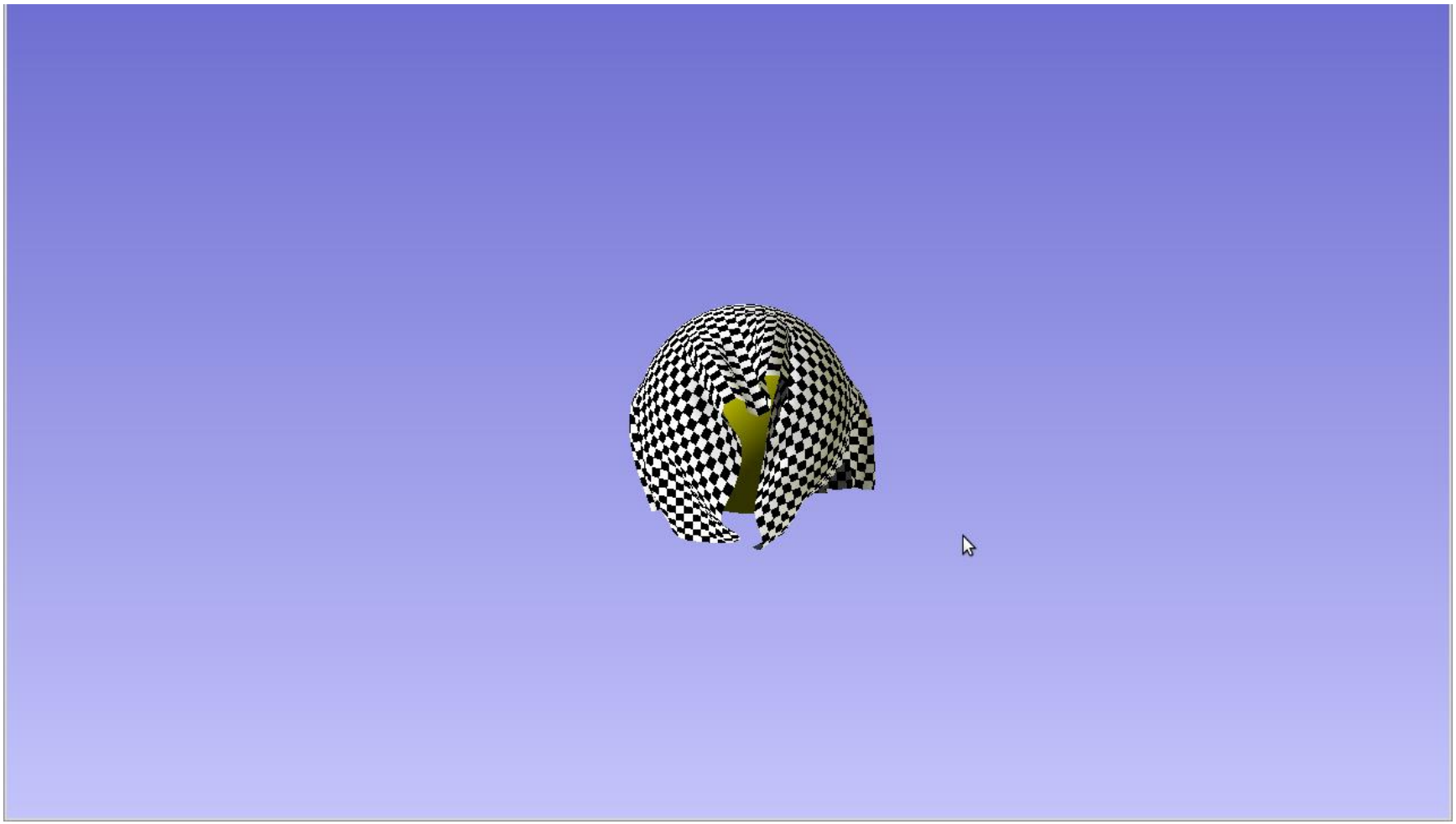
Closeup: Cloth falling onto a stationary sphere



Closeup: Cloth falling onto a stationary sphere



Closeup: Cloth falling onto a stationary sphere



Collision detection with a sphere

Take advantage of the simplistic geometry of a sphere. Every point on the surface of the sphere is equidistant from the sphere center.

We measure the euclidean distance from each particle to the sphere's surface.

Remember: you should subtract the radius of your sphere because we do not want the particle to penetrate the sphere.

Collision detection: example code

```
void compute_ball_collision (particle_node *particle_list_head, vec3 ball_center, float
ball_radius)
{
  particle_node *particle_element;
  int x, y;
  float length_particle_to_ball;
  vec3 particle_to_ball, particle_to_ball_normalized;

  particle_element = particle_list_head;
  for (y = 0; y < NUM_PARTICLES_HEIGHT; y++)
  {
    for (x = 0; x < NUM_PARTICLES_WIDTH; x++)
    {
      particle_to_ball.x = particle_element->pos.x - ball_center.x;
      particle_to_ball.y = particle_element->pos.y - ball_center.y;
      particle_to_ball.z = particle_element->pos.z - ball_center.z;
      length_particle_to_ball = compute_vector_magnitude (particle_to_ball);
      if (length_particle_to_ball < ball_radius) // if the particle is inside the ball
      {
        particle_to_ball_normalized = compute_normalized_vector (particle_to_ball);
        particle_to_ball_normalized.x = particle_to_ball_normalized.x * (ball_radius -
length_particle_to_ball);
        particle_to_ball_normalized.y = particle_to_ball_normalized.y * (ball_radius -
length_particle_to_ball);
        particle_to_ball_normalized.z = particle_to_ball_normalized.z * (ball_radius -
length_particle_to_ball);
        particle_offset_position (particle_element, particle_to_ball_normalized); // project
the particle on the surface of the ball
      }
      particle_element++;
    }
  }
}
```