Московский Государственный Университет им. М.В. Ломоносова
факультет Вычислительной математики и кибернетики
кафедра Суперкомпьютеров и Квантовой информатики

Возможности пакета TaiChi для математического моделирования на современных вычислительных архитектурах

Выполнила:
студентка группы 538, Ши Хуэй

Руководитель:
доцент кафедры СКИ, к.т.н., Русол А.В.

Numpy vs Taichi

```python
for j in range(n - 1):
    dx[j] = x[j + 1] - x[j] - l

for j in range(n):

    left = np.array([0.0, 0.0, 0.0])
    right = np.array([0.0, 0.0, 0.0])
    contact = np.array([0.0, 0.0, 0.0])

    if j > 0:
        left = -C * dx[j - 1]
    if j < n - 1:
        right = C * dx[j]
    if j == 0 and x[j][1] < 0.0:
        contact = -Cs * x[j]

    F[j] = left + right - B * v[j] + contact - m * np.array([0, 9.8, 0])

for j in range(n):
    a = F[j] / m
    v[j] = v[j] + a * dt
    x[j] = x[j] + v[j] * dt
```

```python
@ti.kernel
def substep():
    for j in range(n - 1):
        dx[j] = x[j + 1] - x[j] - ti.Vector([0, l, 0])

    for j in range(n):

        left = ti.Vector([0.0, 0.0, 0.0])
        right = ti.Vector([0.0, 0.0, 0.0])
        contact = ti.Vector([0.0, 0.0, 0.0])

        if j > 0:
            left = -C * dx[j - 1]
        if j < n - 1:
            right = C * dx[j]
        if j == 0 and x[j][1] < 0.0:
            contact = -Cs * x[j]

        F[j] = left + right - B * v[j] + contact - m * ti.Vector([0, 9.8, 0])

    for j in range(n):
        a = F[j] / m
        v[j] = v[j] + a * dt
        x[j] = x[j] + v[j] * dt
```

Метод Эйлера

$$\mathbf{v}_{ij}(t+\Delta t) = \mathbf{v}_{ij}(t) + \frac{\mathbf{F}_{ij}(t)}{m} \cdot \Delta t$$
$$\mathbf{r}_{ij}(t+\Delta t) = \mathbf{r}_{ij}(t) + \mathbf{v}_{ij}(t) \cdot \Delta t$$

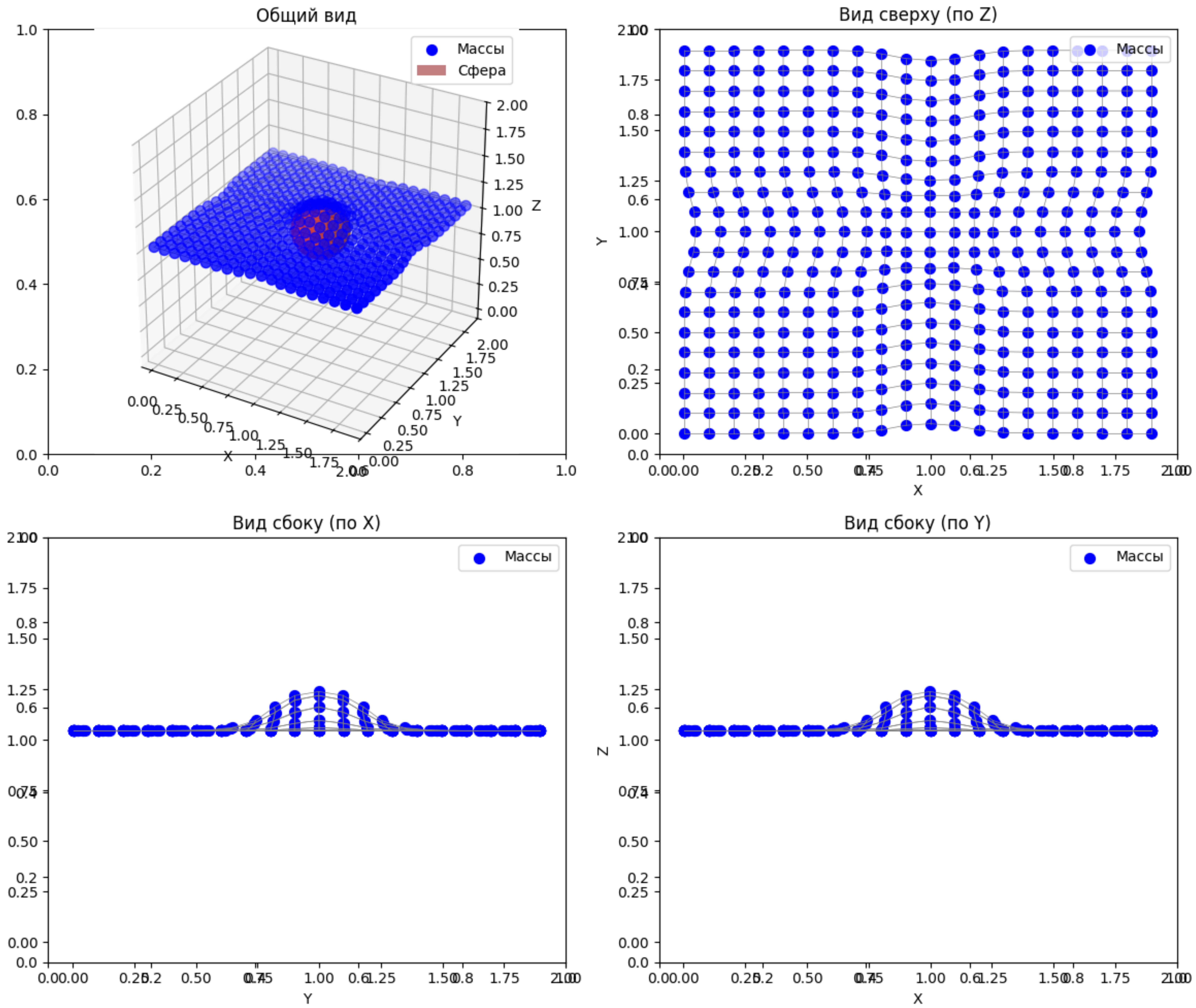
Метод Верле

$$\mathbf{v}_{ij}(t) = \frac{\mathbf{r}_{ij}(t+\Delta t) - \mathbf{r}_{ij}(t-\Delta t)}{2\Delta t}$$
$$\mathbf{r}_{ij}(t+\Delta t) = 2\mathbf{r}_{ij}(t) - \mathbf{r}_{ij}(t-\Delta t) + \frac{\mathbf{F}_{ij}^{\text{total}}(t)}{m}\Delta t^2$$

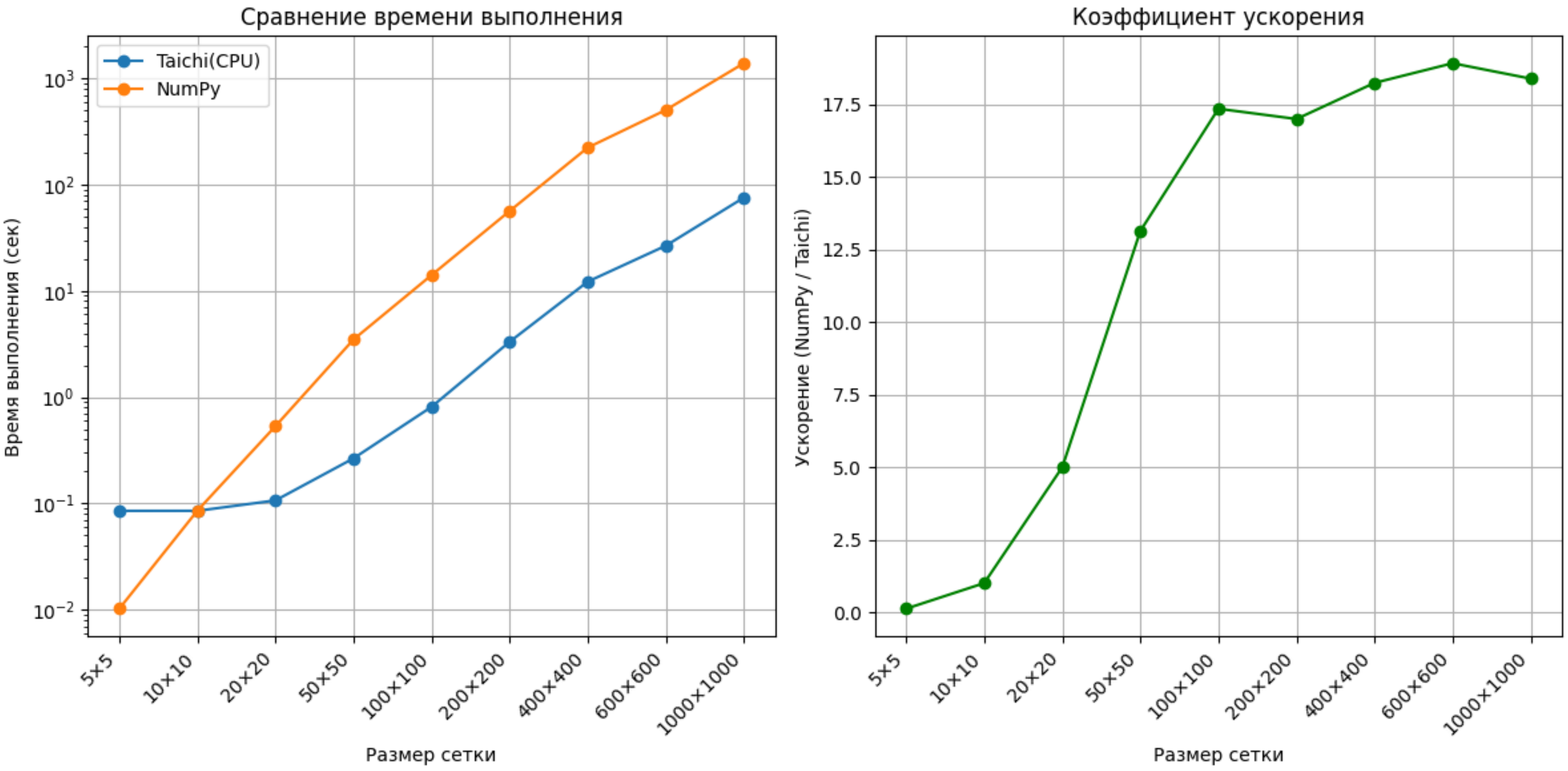
```python
def compute_forces():
    for i in range(N):
        for j in range(M):
            forces[i, j, 2] -= m * g
    for i in range(N):
        for j in range(M):
            r = x[i, j] - sphere_center
            dist = np.linalg.norm(r)
            if dist < R:
                normal = r / dist
                penetration = R - dist
                relative_velocity = np.dot(v[i, j], normal)
                forces[i, j] += Cs * penetration * normal
                        + Bs * relative_velocity * normal
                for _ in range(5):
                    r = x[i, j] - sphere_center
                    dist = np.linalg.norm(r)
                    if dist < R:
                        normal = r / dist
                        penetration = R - dist
                        x[i, j] += penetration * normal
                normal = r / dist
                v_normal = np.dot(v[i, j], normal) * normal
                v_tangent = v[i, j] - v_normal
                v[i, j] = v_tangent
```

```python
@ti.kernel
def compute_forces():
    for i in range(N):
        for j in range(M):
            forces[i, j] = [0, 0, - m * g]
    for i in range(N):
        for j in range(M):
            r = x[i, j] - sphere_center
            dist = ti.sqrt(r.x ** 2 + r.y ** 2 + r.z ** 2)
            if dist < R:
                normal = r / dist
                penetration = R - dist
                relative_velocity = v[i,j] * normal
                forces[i, j] += Cs * penetration * normal
                        + Bs * relative_velocity * normal
                for _ in range(5):
                    r = x[i, j] - sphere_center
                    dist = ti.sqrt(r.x ** 2 + r.y ** 2 + r.z ** 2)
                    if dist < R:
                        normal = r / dist
                        penetration = R - dist
                        x[i, j] += penetration * normal
                normal = r / dist
                v_normal = v[i, j].dot(normal) * normal
                v_tangent = v[i, j] - v_normal
                v[i, j] = v_tangent
```

imageio



Общий вид
Массы
Сфера
X
Y
Z
Вид сверху (по Z)
Массы
X
Y
Вид сбоку (по X)
Массы
Y
Z
Вид сбоку (по Y)
Массы
X
Z



Operating System: Darwin 24.0.0, CPU: Apple M3 Max, Physical Cores: 14, Logical Cores: 14

Спасибо за внимание