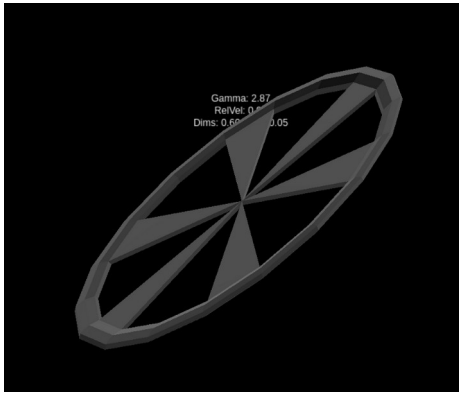
Физика вблизи скорости света: визуализация эффектов СТО

Реализованные эффекты

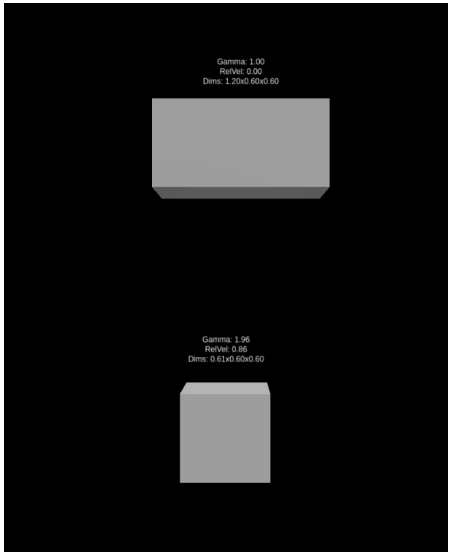
Используя преобразования Лоренца как фундамент для движка, удалось наглядно продемонстрировать ряд релятивистских эффектов:

- Лоренцево замедление времени
- Лоренцево сокращение длины



Эти эффекты можно увидеть на примере следующих экспериментов:

1) Линейные рамеры параллелепипедов
2) Релятивистские гонки
3) Объекты сложной геометрии
4) Город на скоростях света

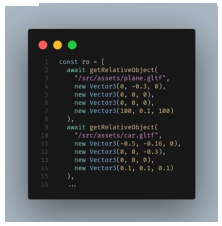


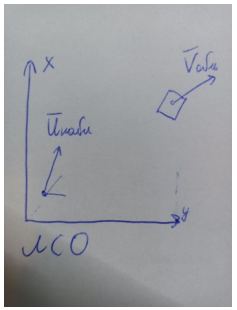
https://mipt-vpv-2024-production.up.railway.app/

Принцип работы

Создание обьектов

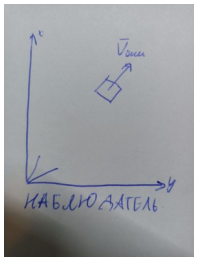
1) Собственные размеры
2) Скорость в лсо





Переход в систему отсчета наблюдателя

1) Расчет относительной скорости для каждого обьекта
2) Расчет центра масс каждого обьекта

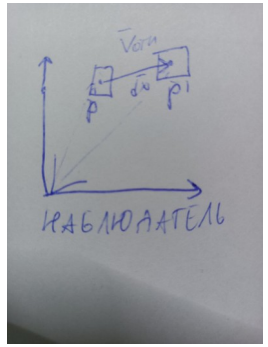


Dt наблюдателя

(каждый тик)

Движение кординаты центра масс обьекта

1) Преобразования Галилея

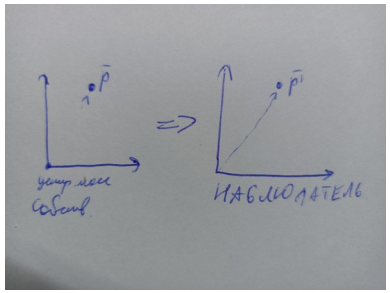


Dt наблюдателя

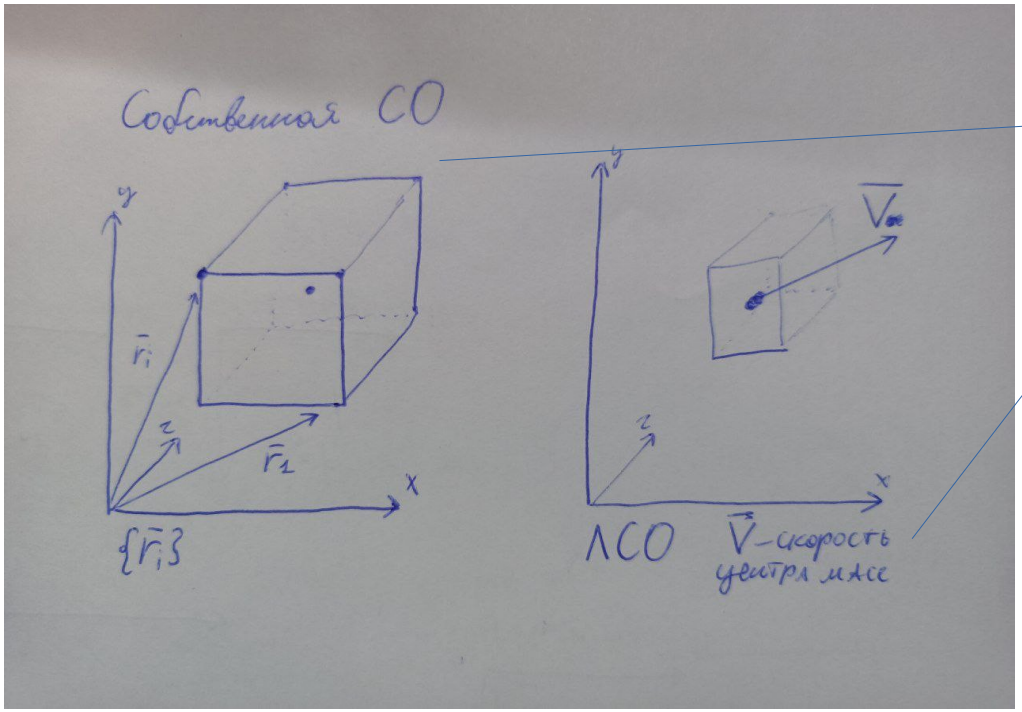
(каждый тик)

Пересчет каждой точки из собственной системы отсчета в систему отсчета наблюдателя

1) Преобразования Лоренса
2) Перенос центра кординат в другую точку (для простоты)



Шаг 1 — Создание обьектов



{ri}

V

```
await getRelativeObject(
  "/src/assets/wheel.gltf",
  new Vector3(0, 0, -1),
  new Vector3(0, 0, 0),
  new Vector3(0.1, 0.1, 0.1),
  new Vector3(Math.PI/2, 0, 0)
),
```

{ri} — собственные кординаты точек обьекта

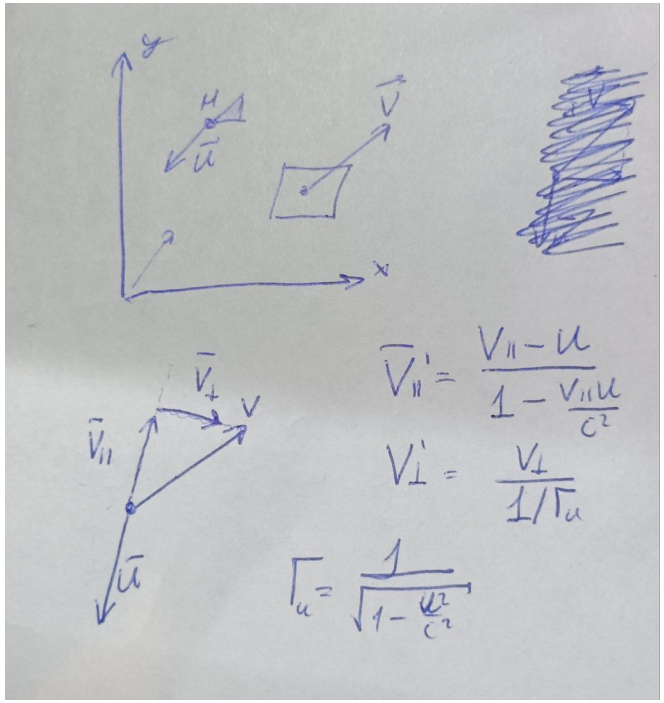
V — скорость в лабараторной системе отсчета

Остальное — позиция, масштаб и угол не очень на что-то влияют, опустим их

Шаг 2 — Переход в систему отсчета наблюдателя

$$V_x' = \frac{dx'}{dt} = \frac{dx - u\,dt}{dt - \frac{u}{c^2}dx} = \frac{V_x - u}{1 - \frac{uV_x}{c^2}}$$

$$V_y' = \frac{dy}{dt'} = \frac{dy}{dt\,\Gamma} = \frac{V_y}{\Gamma}$$



Разложение вектора V по U осям

Vпорал’

Vперп’

Vперп’ + Vпорал’

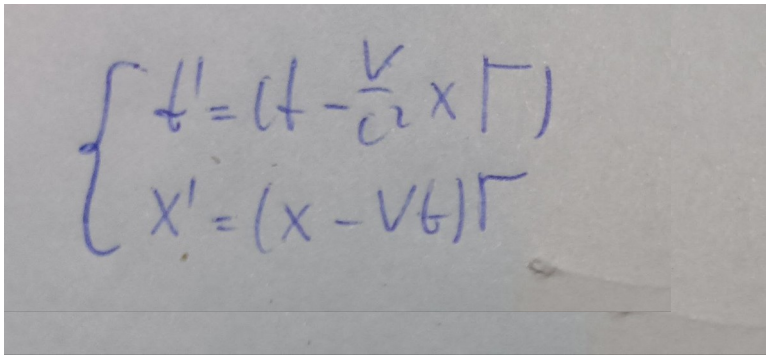
```
getRelativeVelocity(objectVelocity) {
  const u = this.velocity.clone();
  const v = objectVelocity.clone();

  const beta = u.length();
  const direction = u.clone().normalize();

  const vParallel = direction.clone().multiplyScalar(v.dot(direction));
  const vPerpendicular = v.clone().sub(vParallel);

  const parallelSign = Math.sign(v.dot(direction));
  const parallelMagnitude = vParallel.length();
  const transformedParallel = direction.multiplyScalar(
    (parallelMagnitude * parallelSign - beta) /
    (1 - beta * parallelMagnitude * parallelSign)
  );

  const gamma = 1 / Math.sqrt(1 - beta * beta);
  const transformedPerpendicular = vPerpendicular.multiplyScalar(1 / gamma);

  return transformedParallel.add(transformedPerpendicular);
}
```

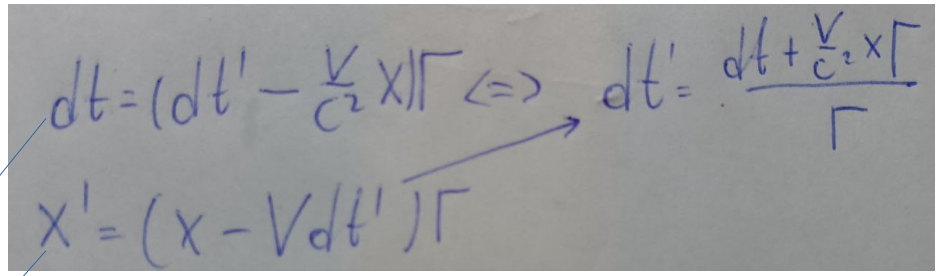
*Расчет центра масс делается как и везде

Шаг 3 — Движение

Перемещение центра масс

```
function update(dt, getObserverVelocity) {
  const localGeometry = this.object.children[0].geometry;
  const projGeometry = this.projObject.children[0].geometry;

  const relVel = getObserverVelocity(this.velocity);
  const gamma = 1 / Math.sqrt(1 - relVel.lengthSq());

  const dx = relVel.clone().multiplyScalar(dt);
  this.position.add(dx);

  const vertices = localGeometry.attributes.position.array;
  for (let i = 0; i < vertices.length; i += 3) {
    const localPoint = new THREE.Vector3(
      vertices[i],
      vertices[i + 1],
      vertices[i + 2]
    );

    const transformedPoint = this.lorentzTransform(
      dt,
      localPoint,
      relVel,
      gamma
    );

    transformedPoint.add(this.position);
    projGeometry.attributes.position.setXYZ(
      i / 3,
      transformedPoint.x,
      transformedPoint.y,
      transformedPoint.z
    );
  }
}

function lorentzTransform(dt, point, vel, gamma) {
  const velMod = vel.length();
  const direction = vel.clone().normalize();

  const xParallel = direction.dot(point);
  const localDt = (dt + velMod * xParallel * gamma) / gamma;

  const xPerpendicular = point
    .clone()
    .sub(direction.clone().multiplyScalar(xParallel));
  const transformedParallel = gamma * (xParallel - velMod * localDt);

  const transformedPoint = xPerpendicular
    .clone()
    .add(direction.multiplyScalar(transformedParallel));

  return transformedPoint;
}
```





Переместим центр масс, и пересчитаем все точки в систему отсчета наблюдателя из собственной

Разложим радиус вектор каждой точки на поралельную относительной скорости часть и перпендикулярную, расчитаем отдельно и сложим.
*Перпендикулярная часть не изменится

Воможности симуляции

- Импорт больших моделей из gltf файлов
- Изменение скорости наблюдателя (моментальное, не ускорение)
- Остановка времени/рендеринга
- Свободное перемещение (во время остановки реднеринга)
- Изменение скорости рендеринга (линейное изменение dt тика)
- Вывод относительной скорости, гамма фактора и линейных (по xyz) размеров

  - Все скорости в расчете на c = 1
  - Все размеры в масштабе 1 = 299 792 458 метров

  *Либо можно предпологать что скорость света в отображаемом мире сапоставима с размерами обьектов

Управление

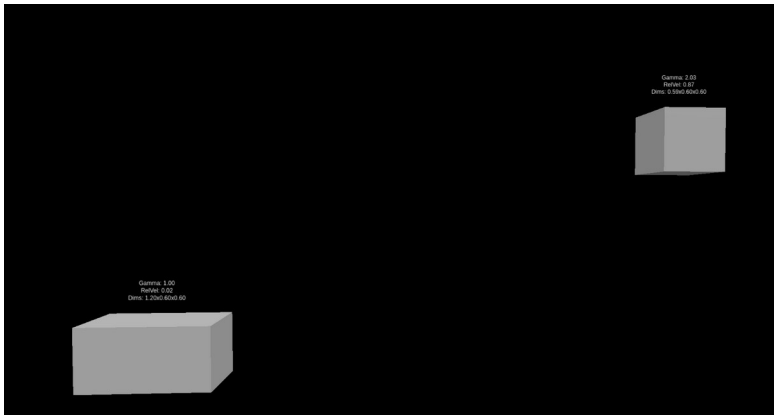
- WASDEQ для изменения скорости наблюдателя при рендеренге(все клавиши привязаны к осям кординат).
- Space для остановки рендеринга/времени.
- WASDEQ для изменения кординаты во время остановки(все клавиши прявязаны к осям кординат).
- Z для сброса скорости наблюдателя при рендеринге.

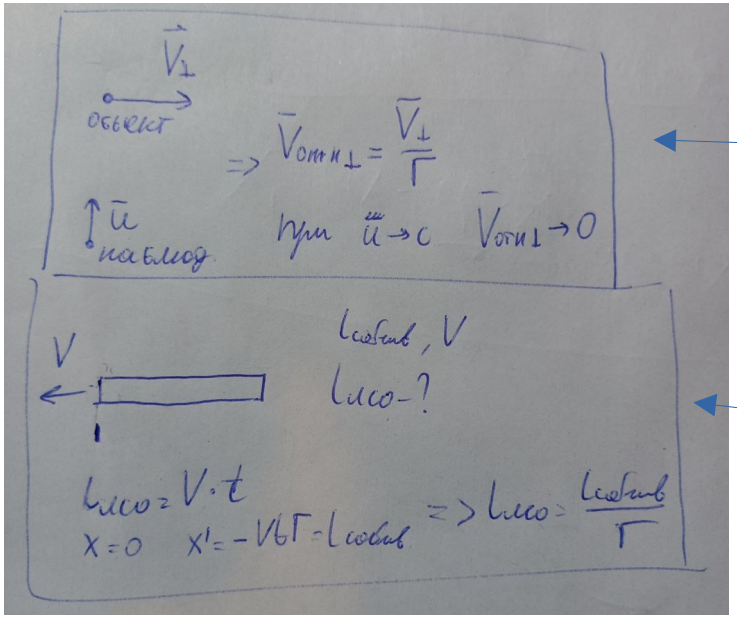
Эксперементы

Линейные рамеры параллелепипедов

Наблюдения:

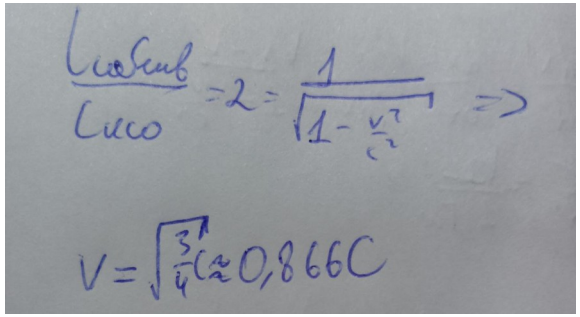
- При движении параллелепипеда со скоростью 0.87c в системе отсчёта наблюдателя он сокращается вдоль направления движения и визуально принимает форму куба. Это проявление Лоренцевого сокращения длины.
- Если перейти в систему отсчёта самого движущегося параллелепипеда, его размеры восстановятся до исходных (собственных), а неподвижный ранее параллелепипед сократится вдоль направления его движения.
- При движении объектов со скоростями, близкими к скорости света, картина для наблюдателя приобретает эффект "застывания": объекты выглядят почти неподвижными из-за релятивистского замедления времени.



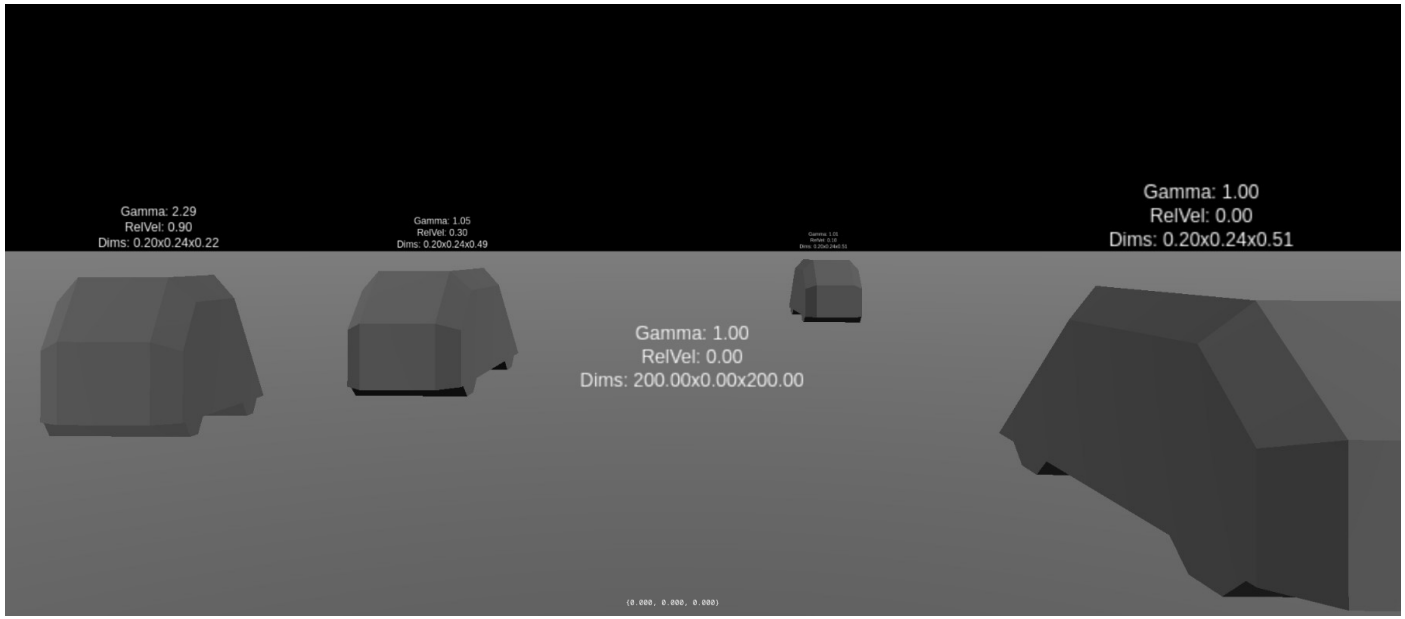


Вывод «застывания»

Вывод Лоренцеого сокращения длины



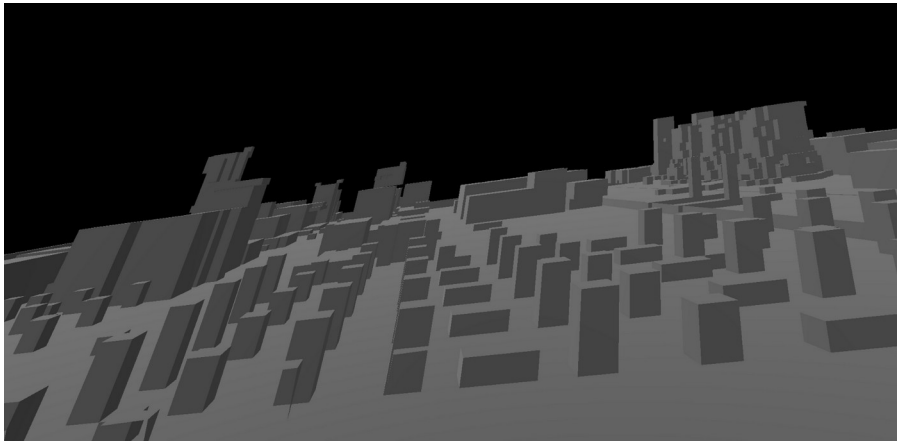
Релятивистские гонки



Наблюдения:

- Все как и в прошлом эксперементе, но большее количество обьектов с различными скоростями.
- Если у наблюдателя будет скорость не только по направлению мышин, то они начнут «вытягиваться» в сторону его движения, что прямо следует из преобразований Лоренца.

Эксперементы 3/4



*Все то же самое, только с более сложной геометрией обьектов.

Вращение

*Надеюсь так можно

Идея

Попробуем еще рассмотреть вращение обьектов вокруг их центров масс, введем для этого вектор w собственной угловой скорости.

Предположительно, так делать можно, ведь обе системы отсчета (наблюдатель и центр масс) двигаются без ускорения, и движение твердого тела можно разложить на кручение вокруг центра масс и движения центра масс.

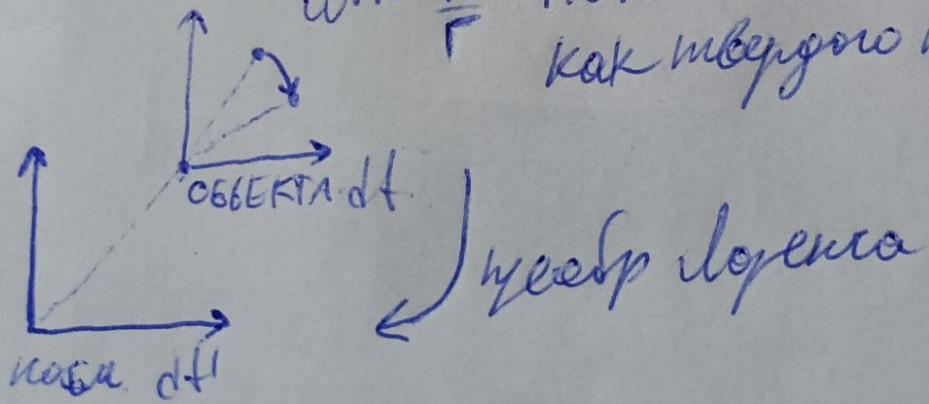
Физические выкладки

$$dt' = (dt - \frac{V}{c^2}(x + dx))\Gamma \qquad dx = dt\, pr_{\bar{x}}\bar{w} \cdot \bar{R}$$

$$dt = \frac{dt' + \frac{V}{c^2} dx \Gamma}{\Gamma(1 - \frac{V}{c^2} pr_{\bar{x}}\bar{w} \cdot \bar{R})}$$

$$X' = ((x + dt \cdot pr_{\bar{x}}\bar{w} \cdot R) - V dt)\Gamma$$

$\bar{w} \cdot \bar{R} \cdot \frac{dt'}{\Gamma}$ – поворот объекта как твердого тела



цикл Лоренца

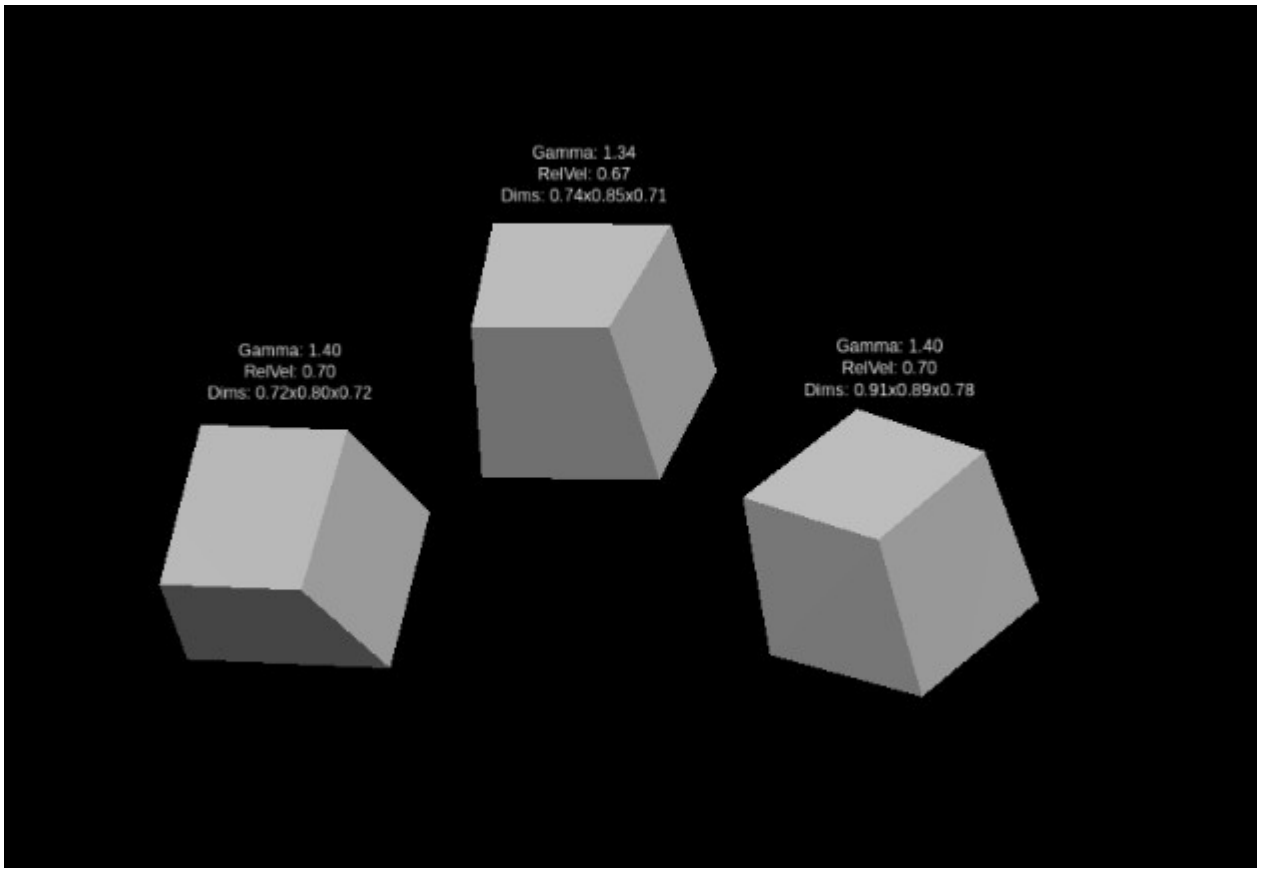
Реализация

```
function update(dt, getObserverVelocity) {
  // Геометрии (точки в системах отсчета)
  const localGeometry = this.object.children[0].geometry;
  const projGeometry = this.projObject.children[0].geometry;

  // Значения
  const relVel = getObserverVelocity(this.velocity);
  const gamma = 1 / Math.sqrt(1 - relVel.lengthSq());

  // Перемещаем центр масс
  const dx = relVel.clone().multiplyScalar(dt);
  this.position.add(dx);

  // Преобразуем точки в глобальную систему наблюдателя с помощью преобразований Лоренца
  const vertices = localGeometry.attributes.position.array;
  for (let i = 0; i < vertices.length; i += 3) {
    const localPoint = new THREE.Vector3(
      vertices[i],
      vertices[i + 1],
      vertices[i + 2]
    );

    const transformedPoint = this.lorentzTransform(
      dt,
      localPoint,
      relVel,
      this.angVelocity,
      gamma
    );

    transformedPoint.add(this.position);
    projGeometry.attributes.position.setXYZ(
      i / 3,
      transformedPoint.x,
      transformedPoint.y,
      transformedPoint.z
    );
  }

  // Обновляем локальное перемещение точек
  const localDt = dt / gamma;
  const localRotAngle = this.angVelocity.length() * localDt;
  const localRotAxis = this.angVelocity.clone().normalize();

  const rotationMatrix = new THREE.Matrix4().makeRotationAxis(
    localRotAxis,
    localRotAngle
  );
  localGeometry.applyMatrix4(rotationMatrix);
}

function lorentzTransform(dt, lastPoint, vel, angVel, gamma) {
  const velMod = vel.length();
  const direction = vel.clone().normalize();

  const lastXParallel = direction.dot(lastPoint);
  const dv = angVel.clone().multiply(lastPoint);
  const localDt =
    (dt + velMod * lastXParallel * gamma) / (gamma * (1 - vel.dot(dv)));
  const point = lastPoint.clone().add(dv.clone().multiplyScalar(localDt));

  const xParallel = direction.dot(point);
  const xPerpendicular = point
    .clone()
    .sub(direction.clone().multiplyScalar(xParallel));

  const transformedParallel = gamma * (xParallel - velMod * localDt);
  const transformedPoint = xPerpendicular
    .clone()
    .add(direction.multiplyScalar(transformedParallel));

  return transformedPoint;
}
```

Усложненное нахождение dt, расчет нового места точки на его основе

Поворот как твердого тела

Эсперементы можно пронаблюдать самостоятельно на сайте, пока все происходящие эффекты не смог полностью доказать, но выглядит достаточно занимательно.



Спасибо за внимание!

https://mipt-vpv-2024-production.up.railway.app/ https://github.com/chififid/MIPT-vpv-2024