OPEN ORIENTED
凹 凸 实 验 室

p2.js – 2D rigid body physics engine

假装国际版本

Why call it p2?

Hmm… Maybe this people is very 2.

“2D rigid body physics engine written in JavaScript. Includes collision detection, contacts, friction, restitution, motors, springs, advanced constraints and various shape types.”

–schteppe

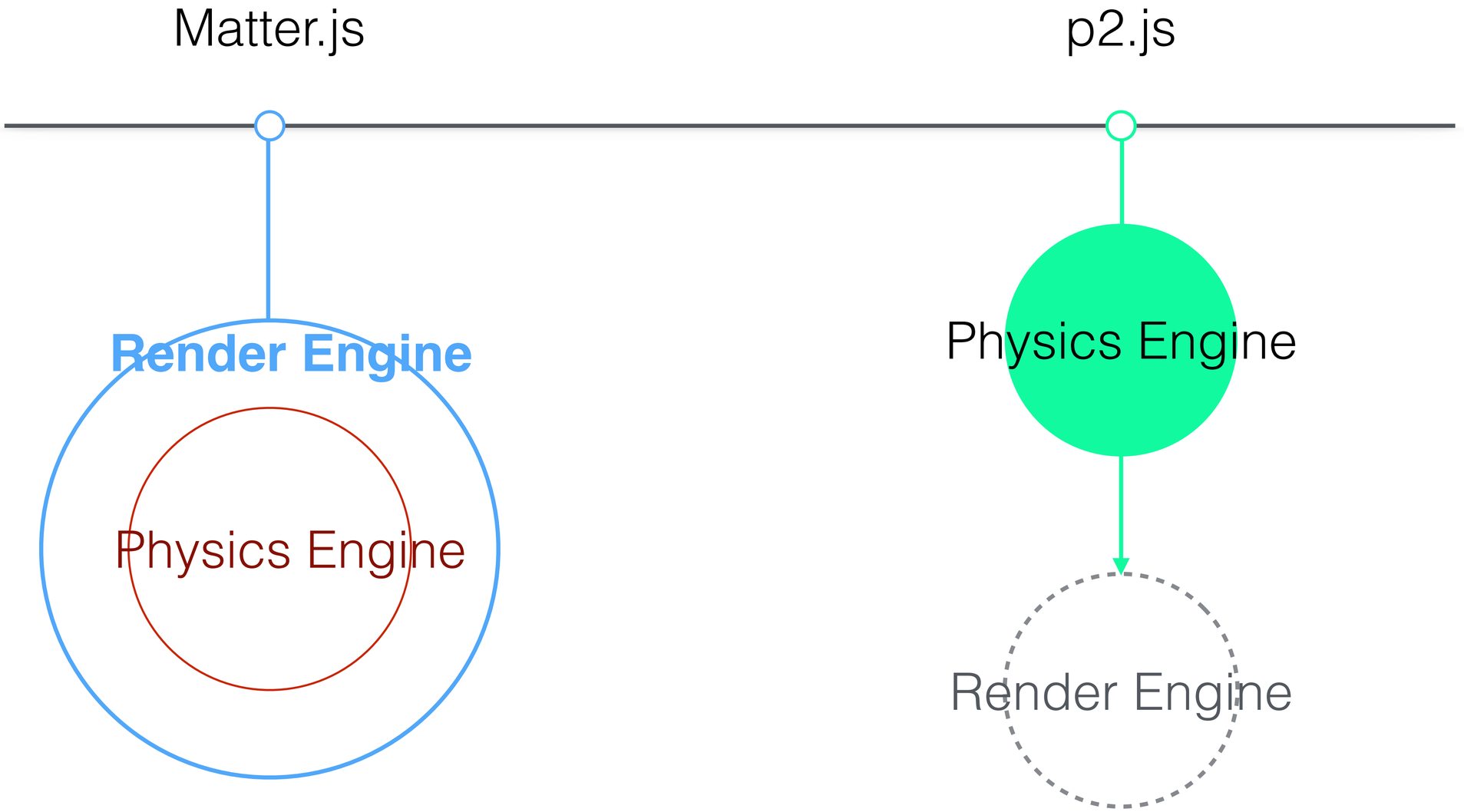
APIs of p2

ContactMaterial
ContactEquation
LinearSpring
Body
OverlapKeeperRecord
Ray
Particle
World
IslandManager
TupleDictionary
Material
Constraint
WheelConstraint
Equation
Narrowphase
Utils
DistanceConstraint
Line
Box
vec2
Spring
IslandNode
LockConstraint
RotationalLockEquation
AABB
Circle
40+APIs
RevoluteConstraint
GSSolver
Plane
EventEmitter
OverlapKeeper
RotationalSpring
Solver
Island
TopDownVehicle
FrictionEquation
SAPBroadphase
RaycastResult
GearConstraint
Heightfield
Capsule
Object pooling utility.
Shape
Convex
RotationalVelocityEquation
PrismaticConstraint
Broadphase
NaiveBroadphase
AngleLockEquation

What can p2 do?

- Buoyancy
- Car
- CCD
- Circle container
- Collision tests
- Compound objects
- Concave objects
- Constraints
- DistanceConstraint
- Fixed rotation
- Fixed XY
- Friction
- Gear constraint
- Heightfield
- Island solver
- Kinematic body
- Lock constraint
- Piston
- Prismatic constraint
- Ragdoll
- Sensor
- Restitution
- Sleep
- Segway
- Sleep
- Springs
- Surface velocity
- Suspension
- Tearable constraints
- TopDownVehicle

Difference between p2 and matter



p2.js may use any render engine

Create.js do rendering

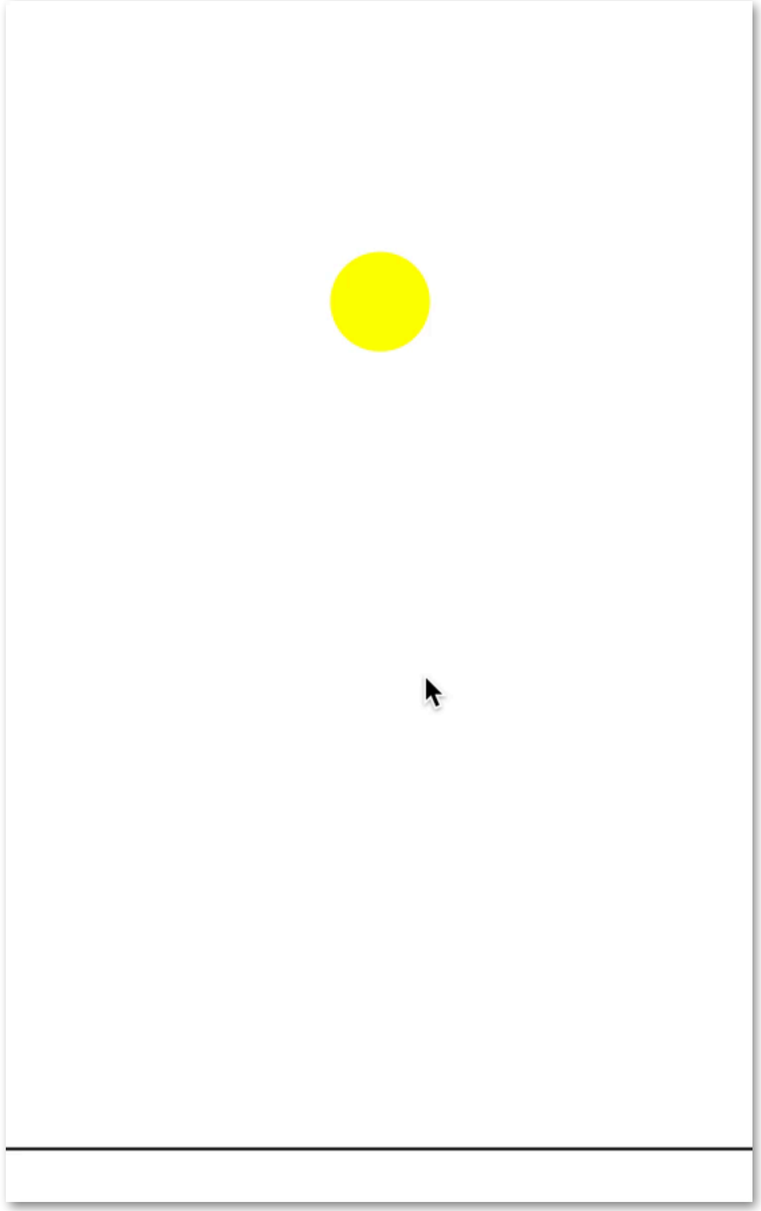

```
// p2 处理物理世界
var world = new p2.World({gravity: [0, -9.8]}), // 创建p2世界
    p2circle = new p2.Body(
        {
            mass: .1,
            position: [0, 6]
        }
    ), // 刚体
    p2circleShape = new p2.Circle({radius: 1}),
    p2plane = new p2.Body({position: [0, -11]});

p2circle.addShape(p2circleShape); // 将形状合并到刚体中
world.addBody(p2circle); // 将刚体添加到世界中
p2plane.addShape(new p2.Plane()); // 添加一个地平线
world.addBody(p2plane);

// createjs 处理渲染
// 创建p2对应的小球
var circle = new createjs.Shape();
circle.graphics.beginFill("#ffff00").drawCircle(0, 0, p2circleShape.radius);
container.addChild(circle);
circle.x = p2circle.position[0];
circle.y = p2circle.position[1];
//创建对应的平面
var plane = new createjs.Shape();
plane.y = -11;
plane.graphics
    .setStrokeStyle(3/ratio)
    .beginStroke("#000")
    .moveTo(-7.5, 0)
    .lineTo(7.5, 0);

container.addChild(plane);

createjs.Ticker.on("tick", function(e) {
    world.step(1/60); // 刷新 p2 世界
    stage.update();
    // 动态刷新状态
    circle.x = p2circle.position[0];
    circle.y = p2circle.position[1];
    circle.rotation = p2circle.angle * 180 / Math.PI;
});
```

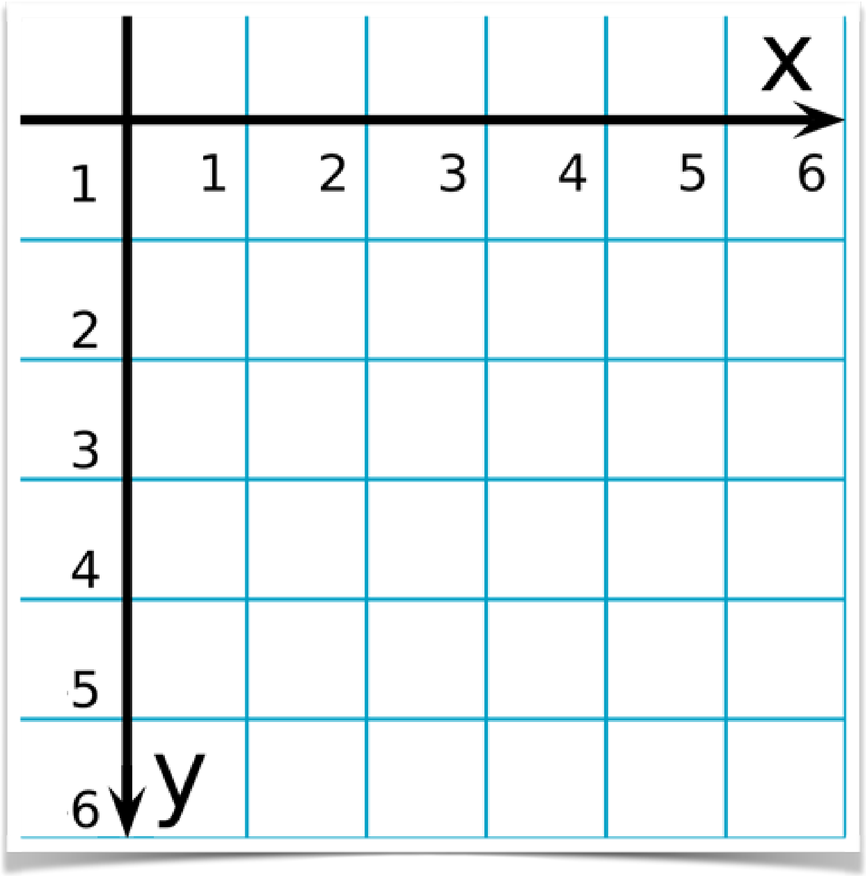


Experience

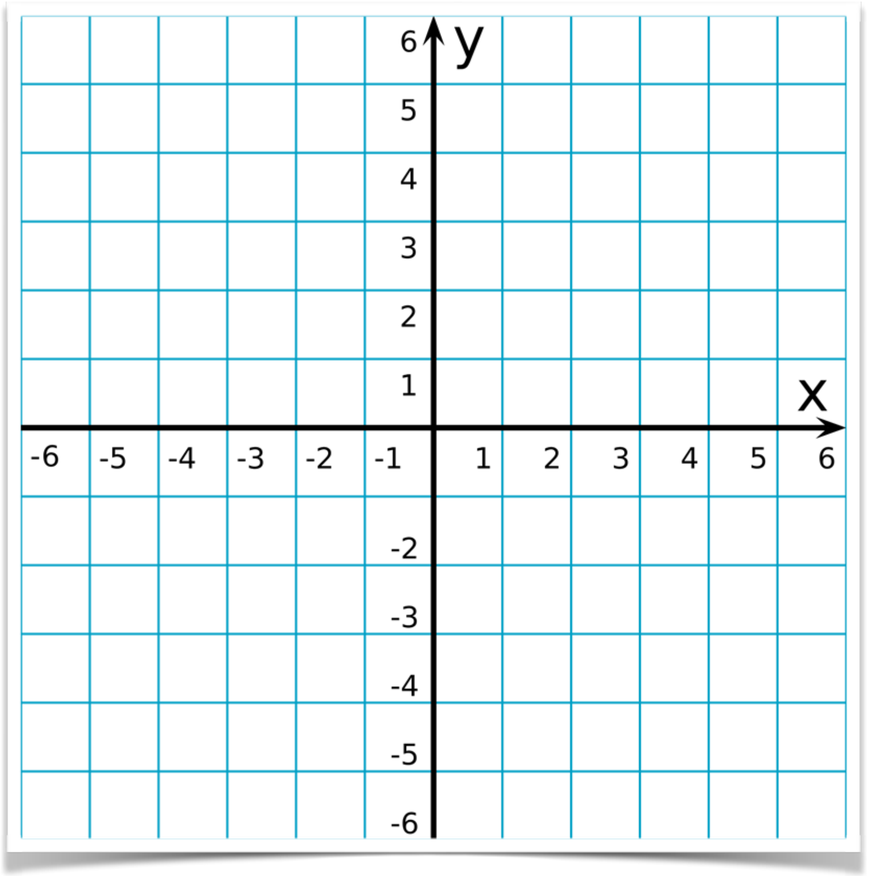
Experience

- Cartesian coordinates
- TimingMode: RAF
- Hollow circle
- properties：angles/position
- force/applyForce
- collisionGroup & collisionMask

Cartesian coordinates



Origin coordinates



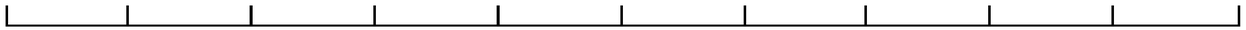
Cartesian coordinates

Cartesian coordinates


```
10  var stage = new createjs.Stage(J_canvas),
11      container = new createjs.Container(),
12      ratio = 50; // p2 的尺寸比率：  1 相当于 50px。
13
14  // 将坐标系统转化为 笛卡尔 坐标
15  stage.x = J_canvas.width / 2;
16  stage.y = J_canvas.height / 2;
17  stage.scaleX = ratio;
18  stage.scaleY = -ratio; // 将Y轴反转
19
```

TimingMode: RAF

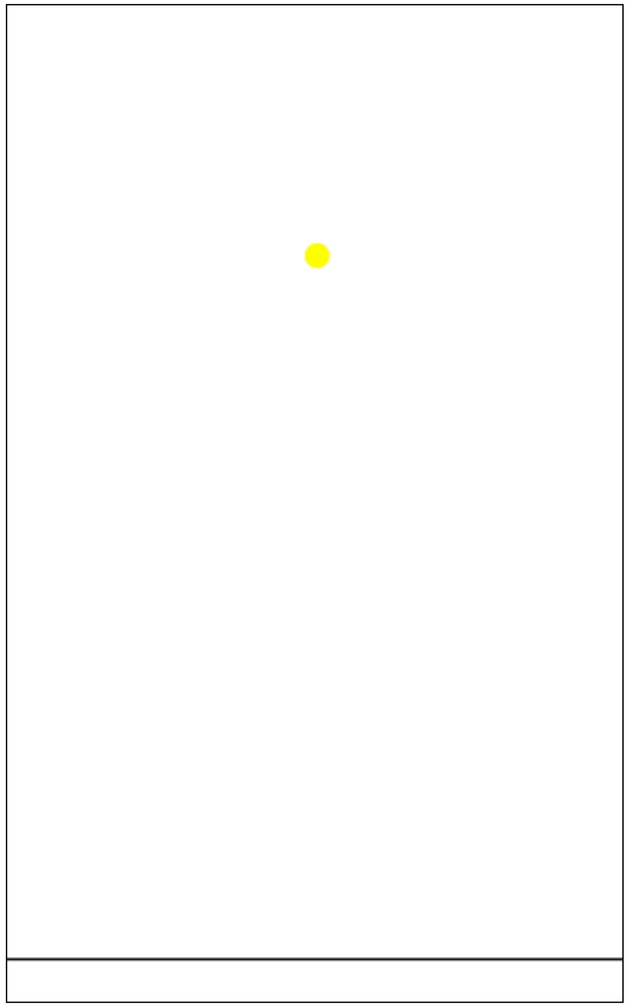
Perfect FPS with RAF:



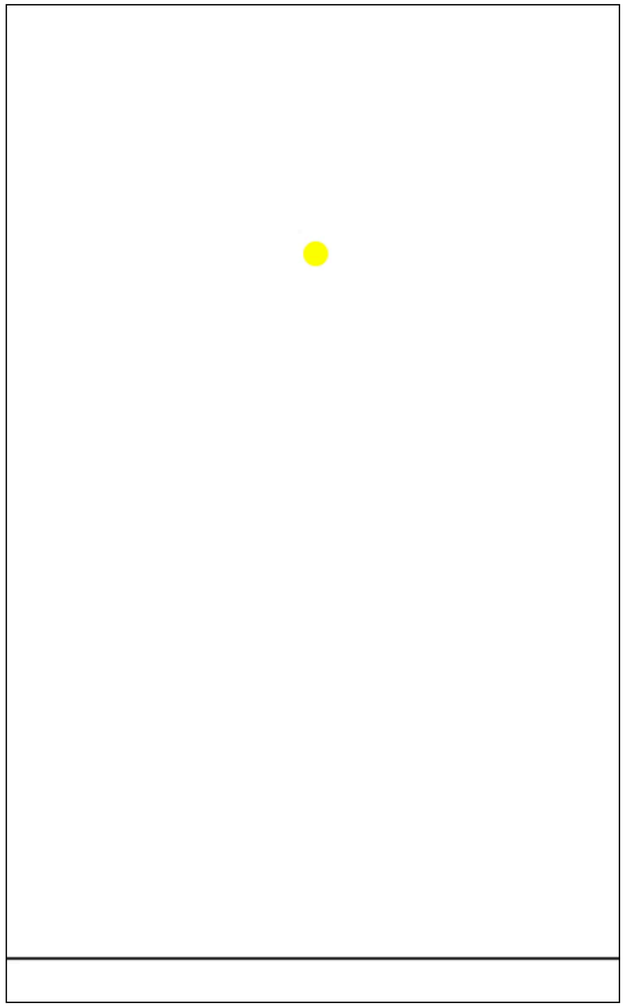
Actully FPS with RAF:

Fixed with interpolation:

TimingMode: RAF



no interpolation



with interpolation

TimingMode: RAF

no interpolation

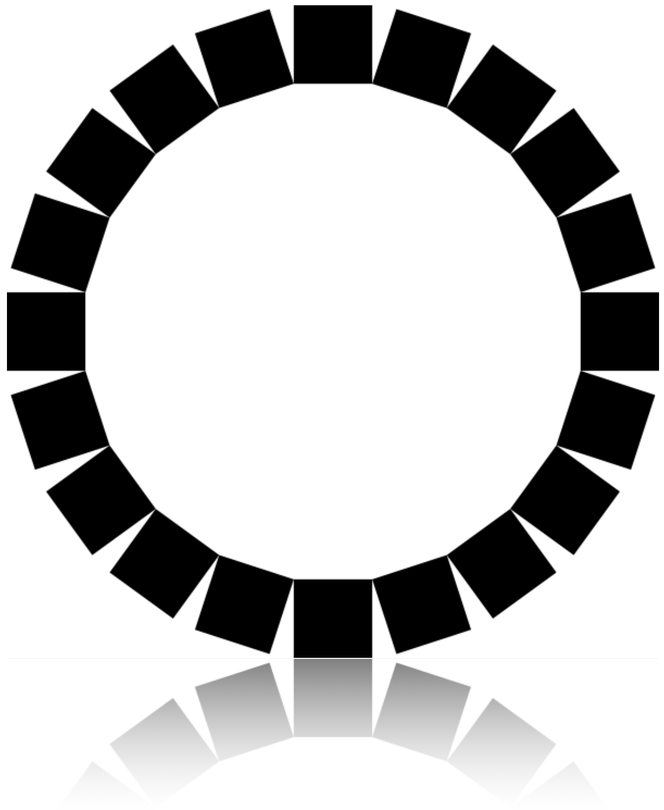
```
createjs.Ticker.setFPS(20); // 用低频帧率模拟低性能
createjs.Ticker.on("tick", function(e) {
    var dt = e.delta / 1000;
    world.step(dt);
    stage.update();
    // 动态刷新状态
    circle.x = p2circle.position[0];
    circle.y = p2circle.position[1];
    circle.rotation = p2circle.angle * 180 / Math.PI;
});
```

with interpolation

```
createjs.Ticker.setFPS(20); // 用低频帧率模拟低性能
createjs.Ticker.on("tick", function(e) {
    var dt = e.delta / 1000, fixedStepTime = 1/60, maxSubSteps = 10;
    world.step(fixedStepTime, dt, maxSubSteps);
    stage.update();
    // 动态刷新状态
    circle.x = p2circle.position[0];
    circle.y = p2circle.position[1];
    circle.rotation = p2circle.angle * 180 / Math.PI;
});
```

Hollow circle

Construct a compound body made of many small wedges shaped in a ring.



properties : angles/position

∵ Body.addShape(shape) == Body.addShape(shape, [0, 0], 0)

∴

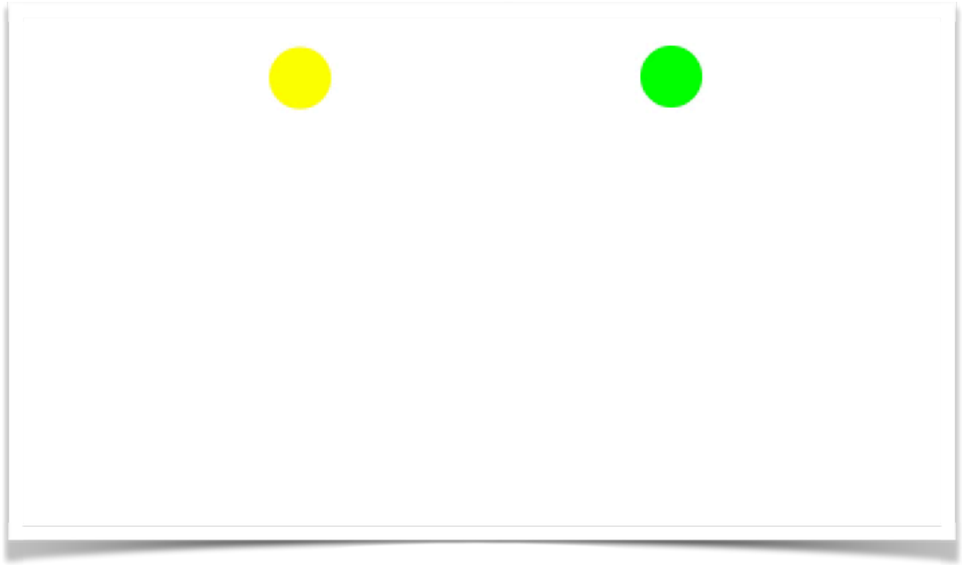
```
shape.position[0] = 2,
shape.position[1] = 2,
shape.angle = Math.PI * .5;
Body.addShape(shape);
```

```
Body.addShape(shape);
shape.position[0] = 2,
shape.position[1] = 2,
shape.angle = Math.PI * .5;
```

```
shape.position[0] = 2, shape.position[1] = 2;
Body.addShape(shape, [2, 2], Math.PI * .5);
```

force/angularForce

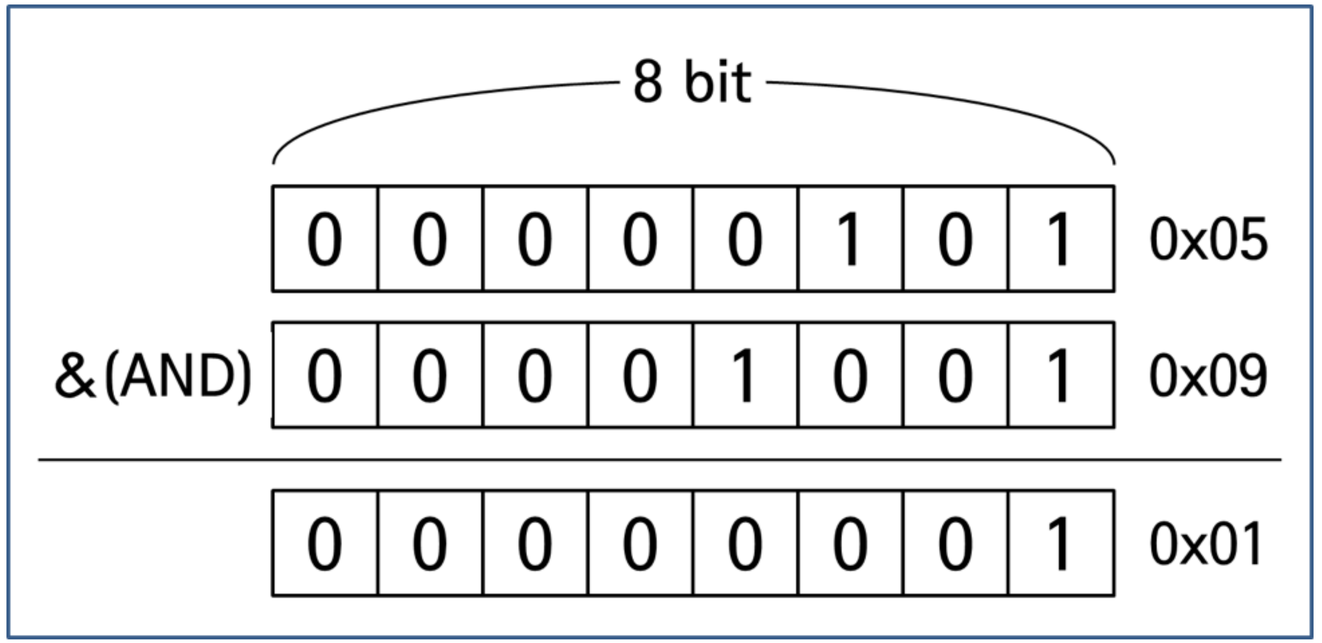
The force acting on the body. Since the body force (and angularForce) will be zeroed after each step, so you need to set the force before each step.



```
p2circle.force[1] = p2circle.mass * world.gravity[1]; // yellow ball
world.on("postStep", function() {
    p2circle2.force[1] = -1 * p2circle2.mass * world.gravity[1]; // green ball
});
```

collisionGroup & collisionMask


```
// How collision check is done
if(shapeA.collisionGroup & shapeB.collisionMask)!=0 && (shapeB.collisionGroup & shapeA.collisionMask)!=0){
      // The shapes will collide
}
```



Reference

- http://schteppe.github.io/p2.js/docs/ — p2官方文档
- https://github.com/schteppe/p2.js/ — p2 的 Github地址
- http://brm.io/matter-js/docs/index.html — matter.js 官方文档

THANKS

FOR YOUR WATCHING

02

OPEN ORIENTED

凹 凸 实 验 室