Week 10: All About P2 Physics in Phaser.js

Phaser and Physics

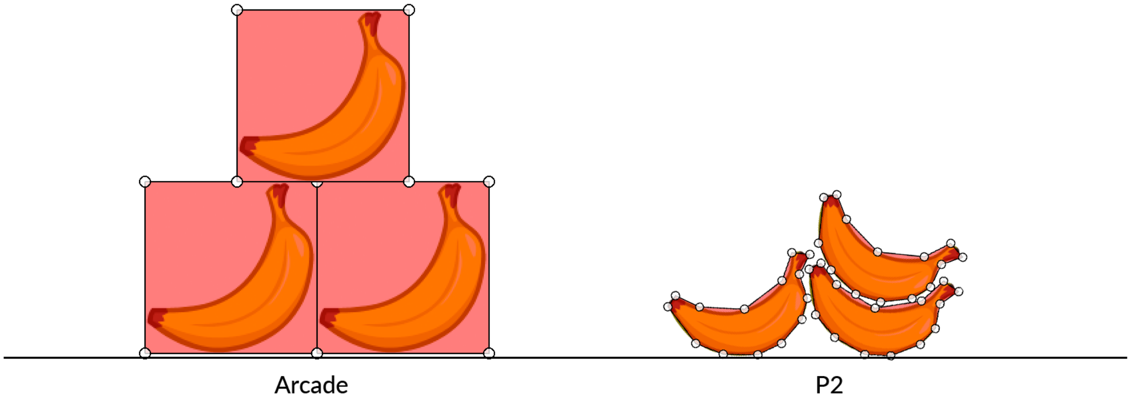
So far we have been using the simplest physics system in the Phaser game engine: Arcade physics. So by now we are all familiar with the **Arcade** physics system. Arcade is the typical physics system used with Phaser and provides a basic physics system that allows for collisions, acceleration, velocity and simple forces like gravity.

In fact, as mentioned previously, the Phaser game engine has **three** physics systems available:

- Arcade Physics
- Ninja Physics
- P2 Physics

This week we dive into the robust simulated physical world of the P2 physics system, where advanced simulations are easily accomplished, thanks to P2. If you are interested in learning about the Ninja physics system, you can **read more about Nina physics here.** **(http://www.joshmorony.com/setting-up-ninja-physics-in-phaser/?_mm=c4f1dZkxk6LD)**

The **P2** physics system allows us to create very complex physical interactions, using things like material properties to determine way forces are transmitted between objects. You can even use simulations of things like springs and pendulums. P2 also allows for complex collisions where the forces are transmitted between colliding objects. Sprite physics body shapes can also be complex. Whereas in the Arcade physics system, the "hitbox" or body for a sprite will always be a simple bounding rectangle.



P2 Sprite Body Shapes

The P2 system allows you to define complex geometric shapes for each sprite's body, making collision detection much more sophisticated. You may define these complex shapes using a dedicated editor, like PhysicsEditor from Code + Web (PhysicsEditor can be found online at **https://www.codeandweb.com/physicseditor** **(https://www.codeandweb.com/physicseditor)**). You can also use dedicated methods available in the P2 system's **Phaser.Physics.P2.Body (https://photonstorm.github.io/phaser-ce/Phaser.Physics.P2.Body.html)** class object value to define body shapes like circles and rectangles that can be combined together as needed.

Working with Body Shapes

Once you enable P2 physics on a sprite, you can apply body shapes to effect the sprite. These shapes include circles, rectangles, lines, polygons as well as other shapes. In the code below, we create a sprite, enable P2 physics, and then apply a circular body shape with a radius of 32 pixels, centered on the sprite. Note that the anchor point is automatically centered for any sprite that has P2 physics applied.

```
var ball = game.add.sprite(spriteX, spriteY, assetkey);
game.physics.p2.enable(ball);
ball.body.clearShapes();
ball.body.setCircle(32);
```

P2 Restitution

The P2 physics system simulates a world where forces can be transmitted between colliding objects. The amount of the force that is transmitted is controlled by the **restitution** property of the P2 physics system. You may set this property to a value between 1.0 and 0.0. When two sprites using P2 collide, the force transmitted will be multiplied by the restitution value. Typically, it is fine to use a value close to 1.0, such as 0.9, so that some force is lost in the collision as it would be in the natural world.

The code below enables the P2 system for your game and then sets the restitution value for the whole P2 physics system in your game.

```
game.physics.startSystem(Phaser.Physics.P2JS);
game.physics.p2.restitution = 0.9;
```

P2 Materials and Contact Materials

The P2 physics system pays attention to the kind of material defined for each sprite using P2. A **Material** value is quite simple in P2, with just a name property associated with the object value in P2. See **Phaser.Physics.P2.Material (https://photonstorm.github.io/phaser-ce/Phaser.Physics.P2.Material.html)** documentation for more information. So defining a material and assigning a sprite body that material is simple.

Things get interesting when sprites collide that each have a material assigned to them. The P2 physics system uses the concept of a Contact Material to define how the physics properties of two colliding materials behave.

The code below defines two Materials, and a Contact Material to control how physics properties are applied when a collision happens.

```
sprite1 = game.add.sprite(100, 100, 'asset1');
game.physics.p2.enable(sprite1);
var spriteMaterial1 = game.physics.p2.createMaterial('spriteMaterial1', sprite1.body);
sprite2 = game.add.sprite(100, 100, 'asset2');
game.physics.p2.enable(sprite2);
var spriteMaterial2 = game.physics.p2.createMaterial('spriteMaterial2', sprite2.body);
var contactMaterial = game.physics.p2.createContactMaterial(spriteMaterial1,spriteMaterial2);
contactMaterial.friction = 0.3; // Friction for the contact of these two materials
contactMaterial.restitution = 1.0; // Restitution for the contact of these two materials
contactMaterial.stiffness = 1e7; // Stiffness of the resulting ContactEquation
contactMaterial.relaxation = 3; // Relaxation of the resulting ContactEquation
contactMaterial.frictionStiffness = 1e7; // Stiffness of the resulting FrictionEquation
contactMaterial.frictionRelaxation = 3; // Relaxation of the resulting FrictionEquation
contactMaterial.surfaceVelocity = 0; // sprite1 rests on top of sprite2, positive value slides sprite1 to right
```

Diving In...

The P2 system is more complex so that it can simulate the physical world more accurately. There are many tutorials and demos that cover more features of P2 found online, including:

- **https://phaser.io/examples/v2/category/p2-physics** (https://phaser.io/examples/v2/category/p2-physics)
- **https://www.codeandweb.com/physicseditor/tutorials/phaser-p2-physics-example-tutorial** (https://www.codeandweb.com/physicseditor/tutorials/phaser-p2-physics-example-tutorial)
- **https://loonride.com/learn/phaser/p2-physics-bodies** (https://loonride.com/learn/phaser/p2-physics-bodies)

Explore and enjoy!