

### Cheat Sheet

This cheat sheet acts as a guide to the Noob to Master physics engine.

### Behaviour options

|                            |                                            |
|----------------------------|--------------------------------------------|
| boolean vanishOnImpact     | If collision, the object disappears        |
| boolean bounceOnImpact     | If collision, the object bounce            |
| boolean constrainedInFrame | If collision with edge, the object bounce  |
| boolean mouseUse           | The object will move with the mouse        |
| boolean keyboardUse        | The object will move with the keypresses   |
| boolean keepXConstant      | The object will only move along the y axis |

These boolean change how an objects react

### Standard physics objects

|                  |                                                                                                              |
|------------------|--------------------------------------------------------------------------------------------------------------|
| <b>Rectangle</b> | Rect(position on X axis, position on Y axis, width, height, Velocity on X axis, Velocity on Y axis)          |
| <b>Circle</b>    | Circle(position on X axis, position on Y axis, radius of the circle, Velocity on X axis, Velocity on Y axis) |

### Add and physics object to the engine

```
PhysicsEngine PE = new PhysicsEngine();

public Pong() {
    p = new Player (width-30, mouseY, 10,
    rectSize, 0, 0);
    c = new Circle(x, y, diam, 5, 5);
    PE.add(p);
    PE.add(c);
}
```

To make your object act according to the physics engine, you have to add it to the physics engine. Above we add a custom physic object called player, and a standart circle to the physics engine.

### Example on custom class that extends Rect

```
public class Player extends Rect {

    //Constructs a Player object and sends the needed
    //parameters to the superclass.
    public Player (float posX, float posY, float
    w, float h, float xVelocity, float yVelocity) {
        super( posX, posY, w, h, xVelocity,
        yVelocity);
        mouseUse = true;
        keepXConstant = true;
    }

    void draw() {
        fill(c);
        rect(pos.x, pos.y, w, h);
        fill(255);
    }
}
```

Here we make a gameobject called player. It is a rectangle, uses the mouse to move and is lock on the X axis.

### Tips for custom classes

You can use an image instead of a rect or circle, just insert it under the draw method in your custom object.

### Ex of customClass, vanish on borderCollision.

```
#1
//Example on custom class that extends Circle.
public class Shot extends Circle {

    //Constructs a Player object and sends the needed
    //parameters to the superclass.
    public Shot(float posX, float posY, float
    radius, float xVelocity, float yVelocity) {
        super( posX, posY, radius, xVelocity,
        yVelocity);
    }

    void draw() {
        fill(c);
        ellipse(pos.x, pos.y, 2*radius, 2*radius);
        fill(255);
    }

    /////// /// /// /// /// /// /////
#2
```



---

By **StuffDK**

[cheatography.com/stuffdk/](https://cheatography.com/stuffdk/)

---

Published 14th December, 2017.

Last updated 17th December, 2017.

Page 1 of 2.

---

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish

Yours!

<https://apollopad.com>

### Ex of customClass, vanish on borderCollision. (cont)

```
> void borderCollisionShot(PhysicsObject PO)
{
    Shot shot = (Shot)PO;
    if (shot.pos.x <= shot.radius)
    {
        if (shot.gotHitX != true)
        {
            objectArray.remove(shot);
            shot.gotHitX = true;
        }
    }
}

//////////
#3
if (PO instanceof Shot)
    borderCollisionShot(PO);
//////////
#2
void borderCollisionShot(PhysicsObject PO)
{
    Shot shot = (Shot)PO;
    if (shot.pos.x <= shot.radius)
    {
        if (shot.gotHitX != true)
        {
            objectArray.remove(shot);
            shot.gotHitX = true;
        }
    }
}

//////////
```

Above, we made a custom class called shot. Its based on the circle class. This is marked #1.

We need to add a borderCollisionShot method, to check if there is a border collision and remove the shot from the Array.

#2 does this. This does only check one of the borders, so you have to make the rest yourself. Look at the borderCollisionCircle to get inspiration.

Then we have to finde were the physics engine checkes border collision and add in #3, so the physics engine will run the method.

### More information

You can find information on all classes and methods, in the readme.txt file in the physics engine folder.



By **StuffDK**  
[cheatography.com/stuffdk/](https://cheatography.com/stuffdk/)

Published 14th December, 2017.  
 Last updated 17th December, 2017.  
 Page 2 of 2.

Sponsored by **ApolloPad.com**  
 Everyone has a novel in them. Finish Yours!  
<https://apollopad.com>