We have the bricks appearing on the screen already, but the game still isn't *that* interesting as the ball goes through them. We need to think about adding collision detection so it can bounce off the bricks and break them.

It's our decision how to implement this, of course, but it can be tough to calculate whether the ball is touching the rectangle or not because there are no helper functions in Canvas for this. For the sake of this tutorial we will do it the easiest way possible. We will check if the center of the ball is colliding with any of the given bricks. This won't give a perfect result every time, and there are much more sophisticated ways to do collision detection, but this will work fine for teaching you the basic concepts.

**A collision detection function**

To kick this all off we want to create a collision detection function that will loop through all the bricks and compare every single brick's position with the ball's coordinates as each frame is drawn. For better readability of the code we will define the b variable for storing the brick object in every loop of the collision detection:

function collisionDetection() {

for(c=0; c<brickColumnCount; c++) {

for(r=0; r<brickRowCount; r++) {

var b = bricks[c][r];

// calculations

}

}

}

If the center of the ball is inside the coordinates of one of our bricks, we'll change the direction of the ball. For the center of the ball to be inside the brick, all four of the following statements need to be true:

* The x position of the ball is greater than the x position of the brick.
* The x position of the ball is less than the x position of the brick plus its width.
* The y position of the ball is greater than the y position of the brick.
* The y position of the ball is less than the y position of the brick plus its height.

Let's write that down in code:

function collisionDetection() {

for(c=0; c<brickColumnCount; c++) {

for(r=0; r<brickRowCount; r++) {

var b = bricks[c][r];

if(x > b.x && x < b.x+brickWidth && y > b.y && y < b.y+brickHeight) {

dy = -dy;

}

}

}

}

Add the above block to your code, below the keyUpHandler() function.

**Making the bricks disappear after they are hit**

The above code will work as desired and the ball changes its direction. The problem is that the bricks are staying where they are. We have to figure out a way to get rid of the ones we've already hit with the ball. We can do that by adding an extra parameter to indicate whether we want to paint each brick on the screen or not. In the part of the code where we initialize the bricks, let's add a status property to each brick object. Update the following part of the code as indicated by the highlighted line:

var bricks = [];

for(c=0; c<brickColumnCount; c++) {

bricks[c] = [];

for(r=0; r<brickRowCount; r++) {

bricks[c][r] = { x: 0, y: 0, status: 1 };

}

}

Next we'll check the value of each brick's status property in the drawBricks() function before drawing it — if status is 1, then draw it, but if it's 0, then it was hit by the ball and we don't want it on the screen anymore. Update your drawBricks() function as follows:

function drawBricks() {

for(c=0; c<brickColumnCount; c++) {

for(r=0; r<brickRowCount; r++) {

if(bricks[c][r].status == 1) {

var brickX = (c\*(brickWidth+brickPadding))+brickOffsetLeft;

var brickY = (r\*(brickHeight+brickPadding))+brickOffsetTop;

bricks[c][r].x = brickX;

bricks[c][r].y = brickY;

ctx.beginPath();

ctx.rect(brickX, brickY, brickWidth, brickHeight);

ctx.fillStyle = "#0095DD";

ctx.fill();

ctx.closePath();

}

}

}

}

**Tracking and updating the status in the collision detection function**

Now we need to involve the brick status property in the collisionDetection() function: if the brick is active (its status is 1) we will check whether the collision happens; if a collision does occur we'll set the status of the given brick to 0 so it won't be painted on the screen. Update yourcollisionDetection() function as indicated below:

function collisionDetection() {

for(c=0; c<brickColumnCount; c++) {

for(r=0; r<brickRowCount; r++) {

var b = bricks[c][r];

if(b.status == 1) {

if(x > b.x && x < b.x+brickWidth && y > b.y && y < b.y+brickHeight) {

dy = -dy;

b.status = 0;

}

}

}

}

}

**Activating our collision detection**

The last thing to do is to add a call to the collisionDetection() function to our main draw()function. Add the following line to the draw() function, just below the drawPaddle() call:

collisionDetection();

**Compare your code**

The collision detection of the ball is now checked on every frame, with every brick. Now we can destroy bricks! :-

**Exercise**: change the color of the ball when it hits the brick.

**Next steps**

We are definitely getting there now; let's move on! In the eighth chapter we will be looking at how to [Track the score and win](https://developer.mozilla.org/en-US/docs/Games/Workflows/Breakout_game_from_scratch/Track_the_score_and_win).