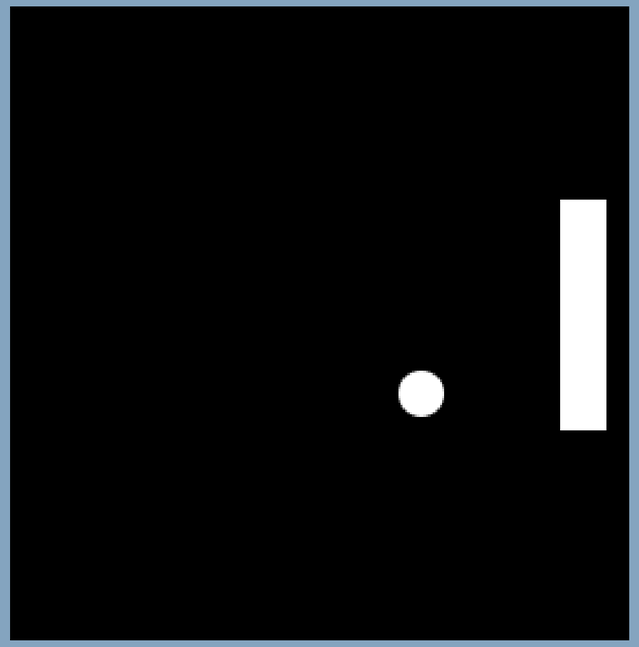
Matemática Aplicada à Multimídia I

PONG

Teste de Colisão



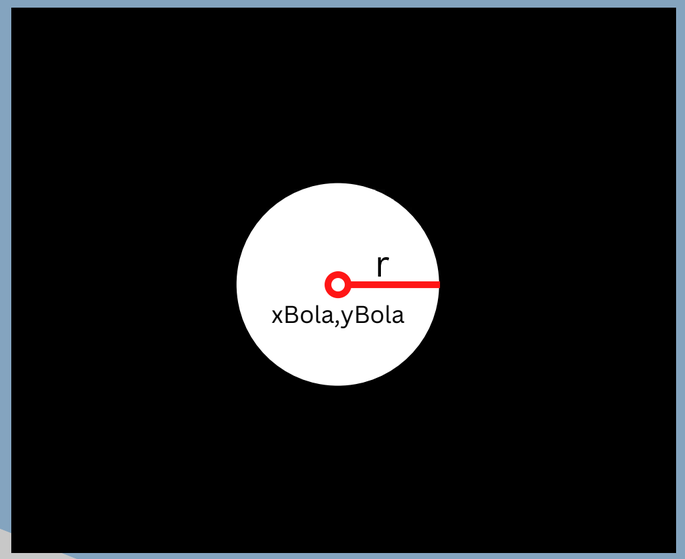
Daniel Cardeal - 541875

Profª - Mara Bonates

Segundo conceitos vistos em aula, temos vários tipos de colisões possíveis para a construção do PONG. E para construirmos essa discussão, usaremos RECT-RECT e PONTO-CIRCULO

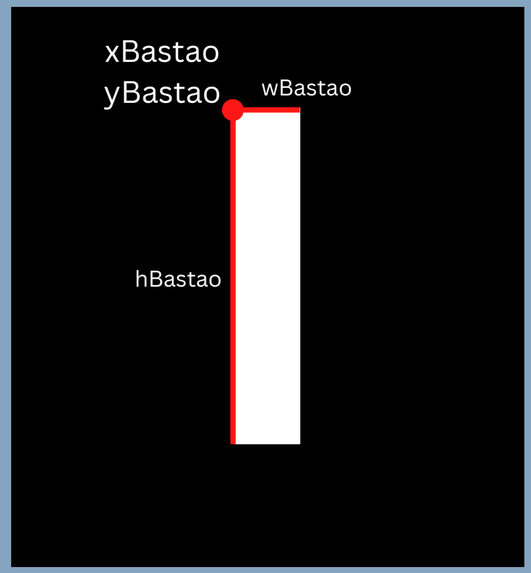
Para discutirmos sobre os diferentes tipos de colisões, precisamos ver qual é o comportamento da bolinha e do bastão:

Bolinha



A bolinha possui um valor **r** de raio e uma posição **xBola** e **yBola**

Bastão

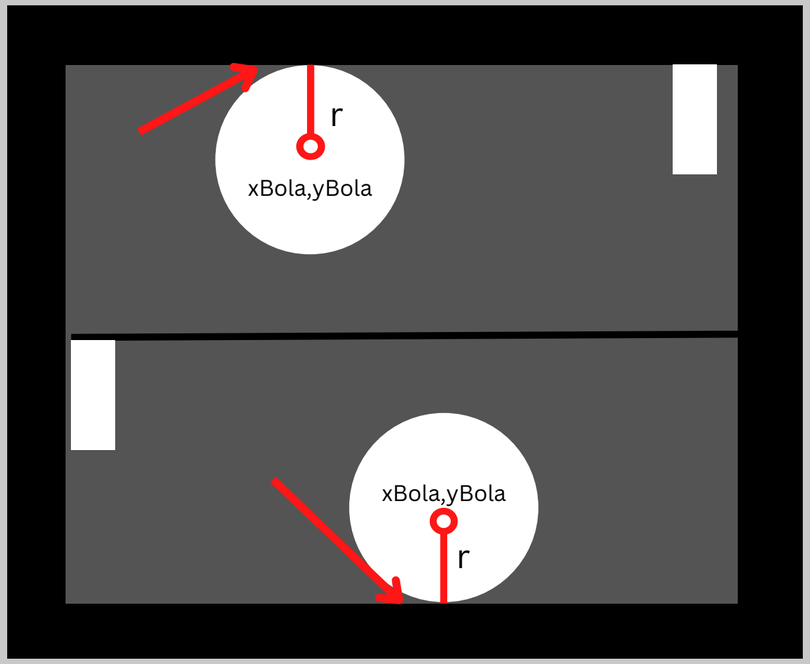


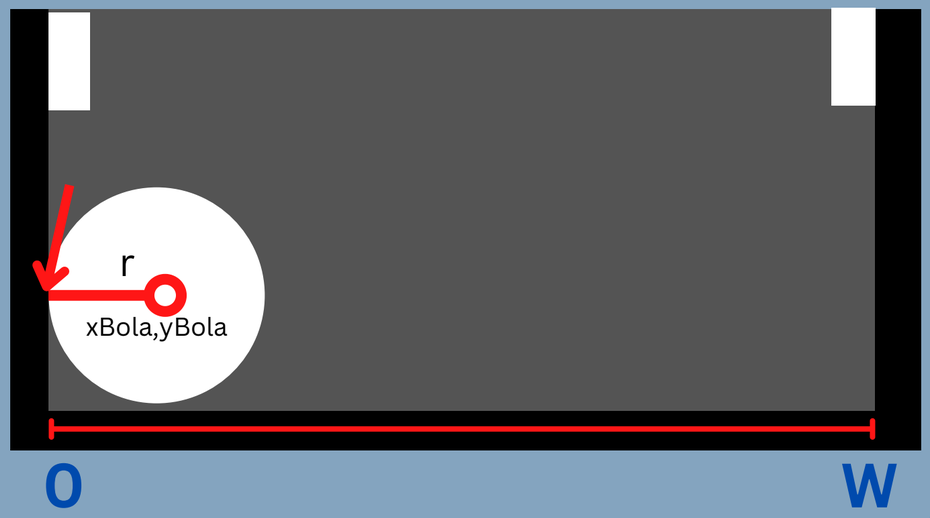
O bastão possui um valor **wBastao** de largura e **hBastao** de altura e uma posição **xBastao** e **yBastao**

O objetivo agora é descobrir como funcionará as colisões.
Vamos analisar primeiro entre a bolinha - parede em diferentes posições

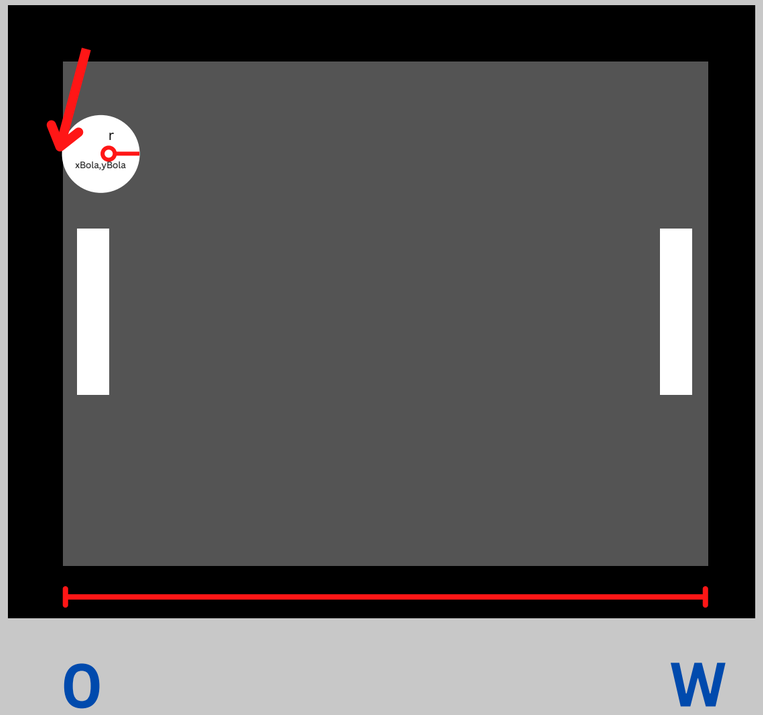
Bolinha-Parede

Tipo de colisão: RECT-RECT





Bolinha-Parede Esquerda



Primeiro vamos testar se ela bateu na esquerda da tela.

Como vamos verificar se a bolinha colidiu com as paredes laterais, consideramos o valor total = W

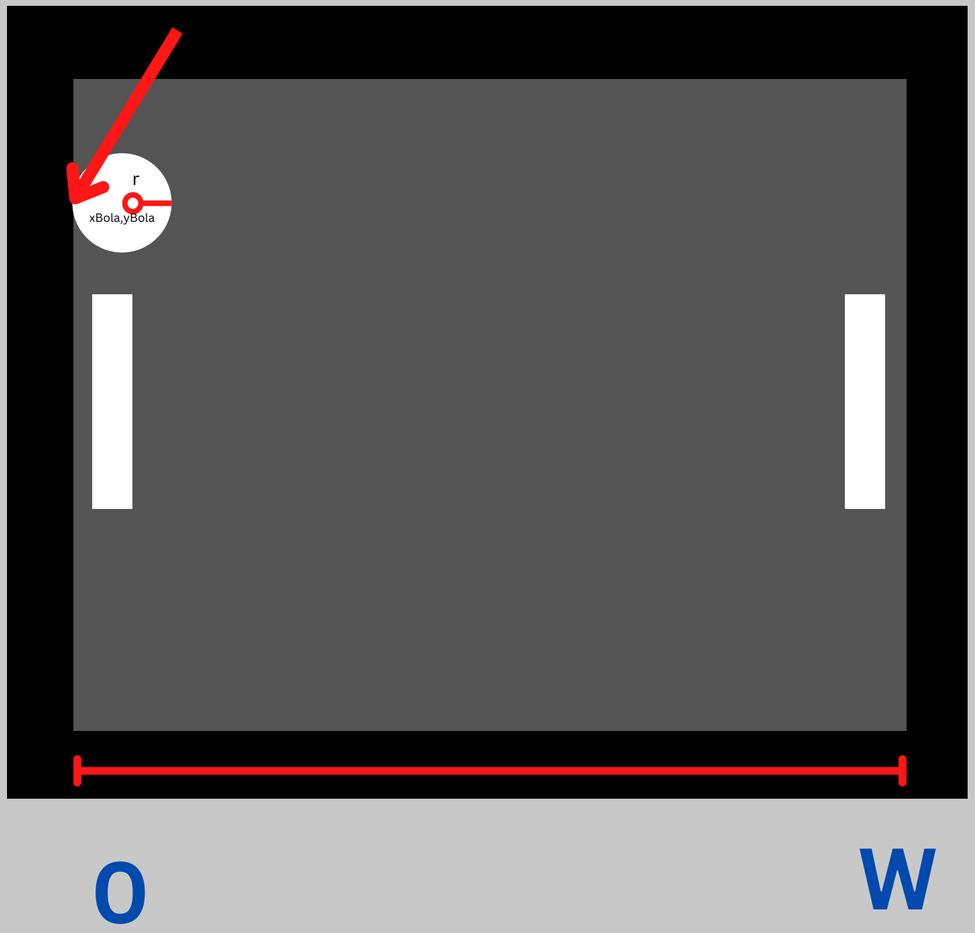
No caso da colisão da esquerda, o W = 0, então basta pegar o xBola e diminuir do raio e verificar se é igual ou menor que 0.

Se for, colidiu.

Bolinha-Parede-Esquerda

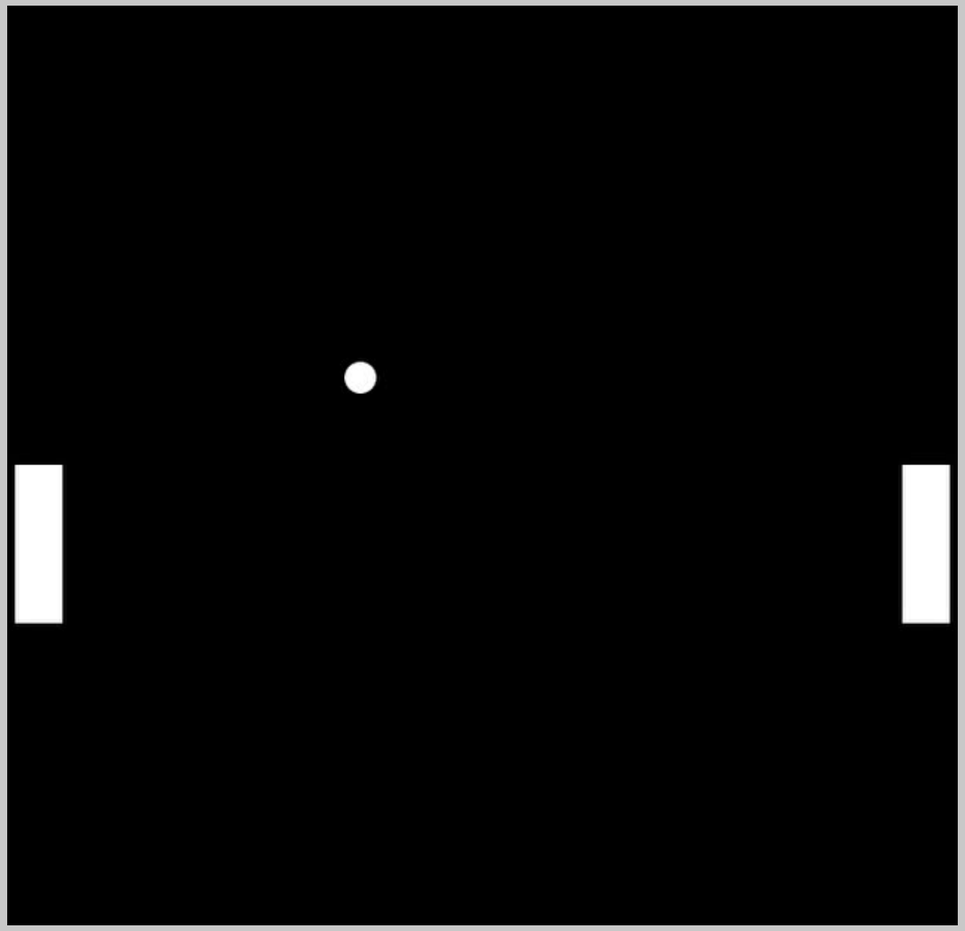
Como a regra do PONG é: Se passar da tela, a bola volta para o centro, então o calculo computacional será dada assim:

```
if(xBola<=0){
XBola=W/2;
yBola=H/2;
}
```



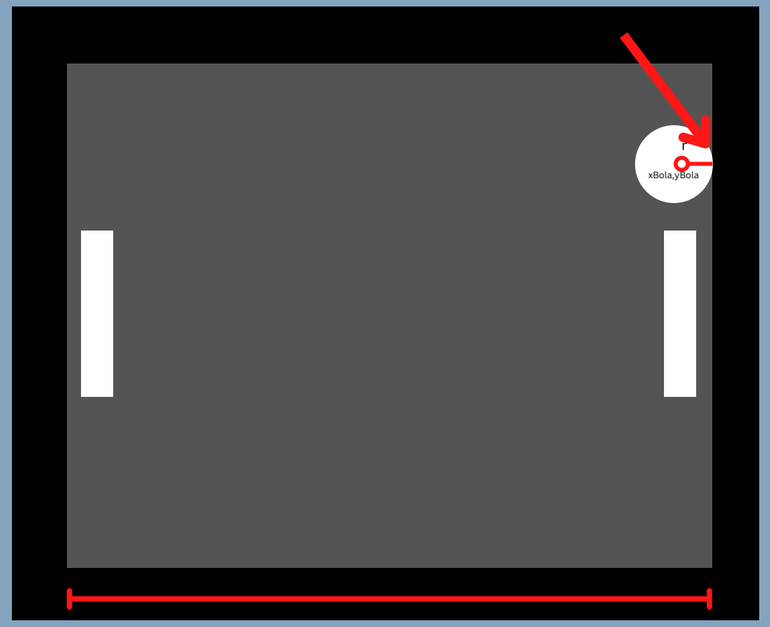
Bolinha-Parede-Esquerda

O resultado esperado é esse do vídeo ao lado:



W

Bolinha-Parede Direita



W

Agora vamos testar se ela bateu na direita da tela

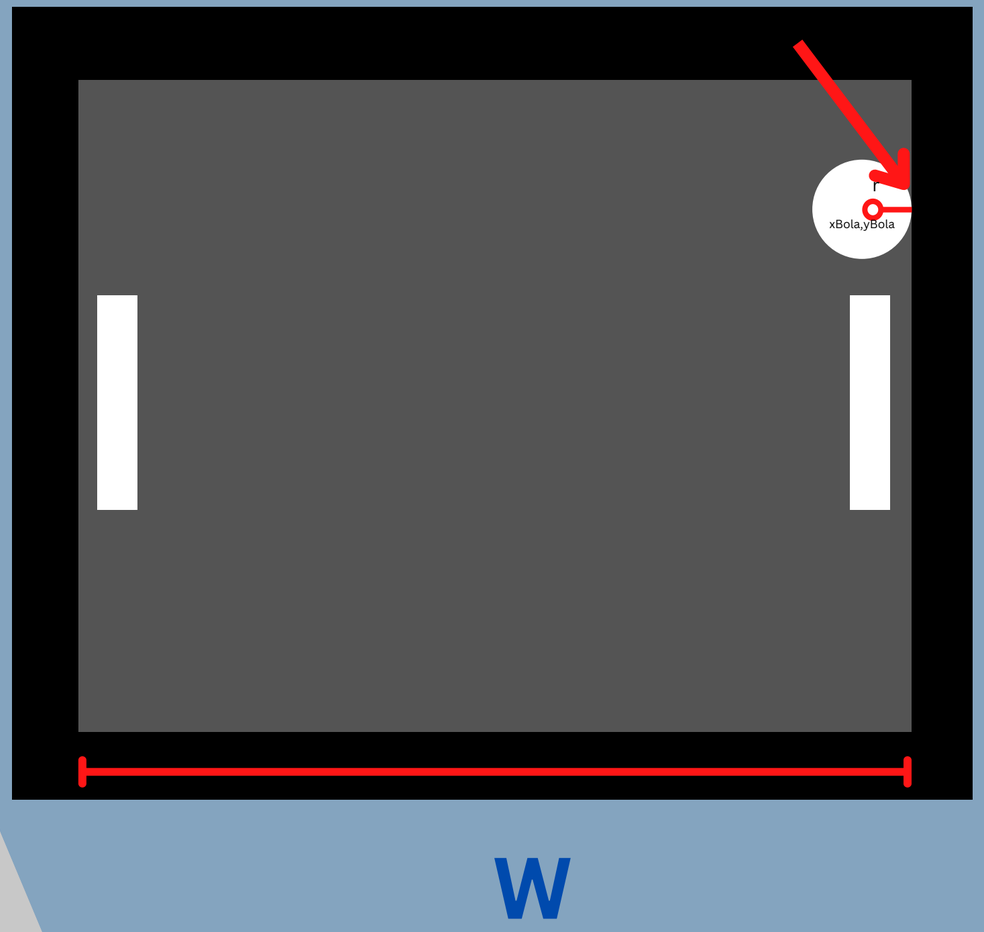
No caso da colisão da Direita, o W vai ser o seu valor total, então basta pegar o xBola somar com o raio e verificar se é maior ou igual que W.

Se for, colidiu.

Bolinha-Parede-Direita

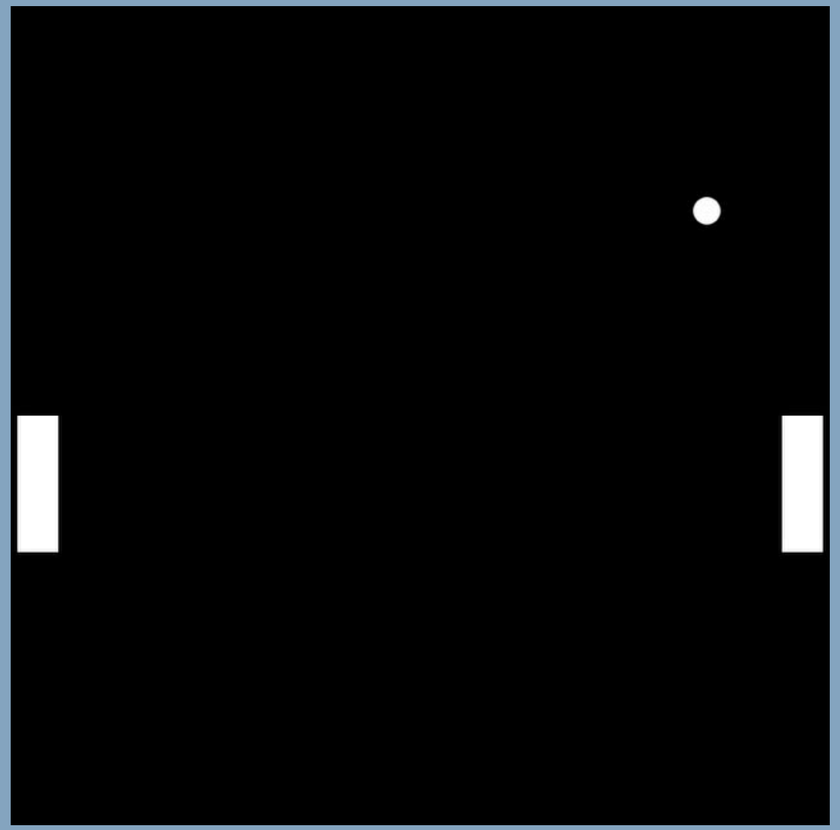
Como a regra do PONG é: Se passar da tela, a bola volta para o centro, então o calculo computacional será dada assim:

```
if(xBola>=W){
XBola=W/2;
yBola=H/2;
}
```

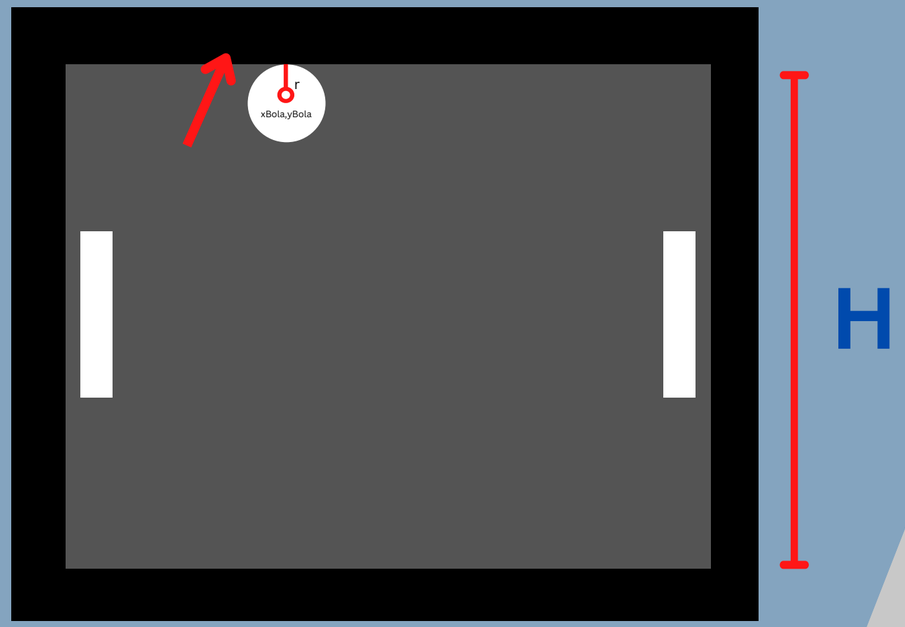


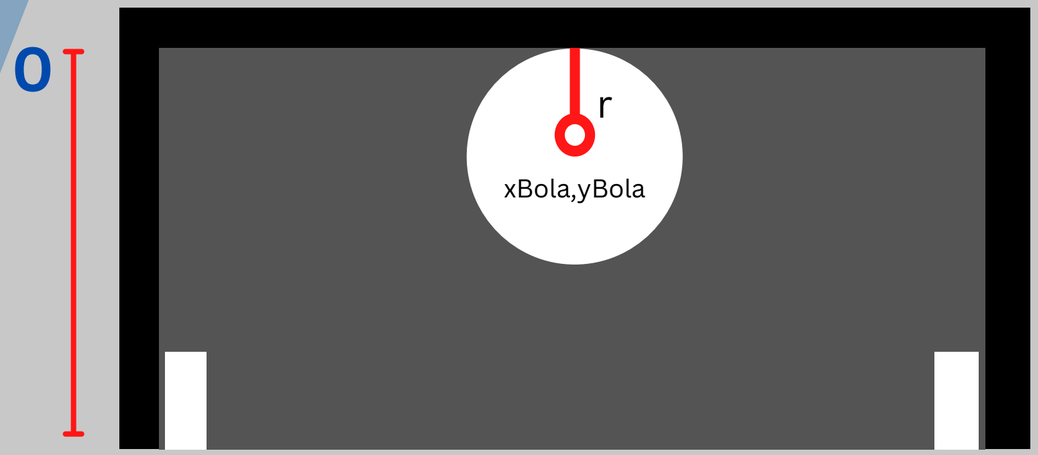
Bolinha-Parede-Direita

O resultado esperado é esse do vídeo ao lado:



Bolinha-Parede de Cima





Agora vamos testar se ela bateu em cima e embaixo, e para isso consideramos a altura da tela de H

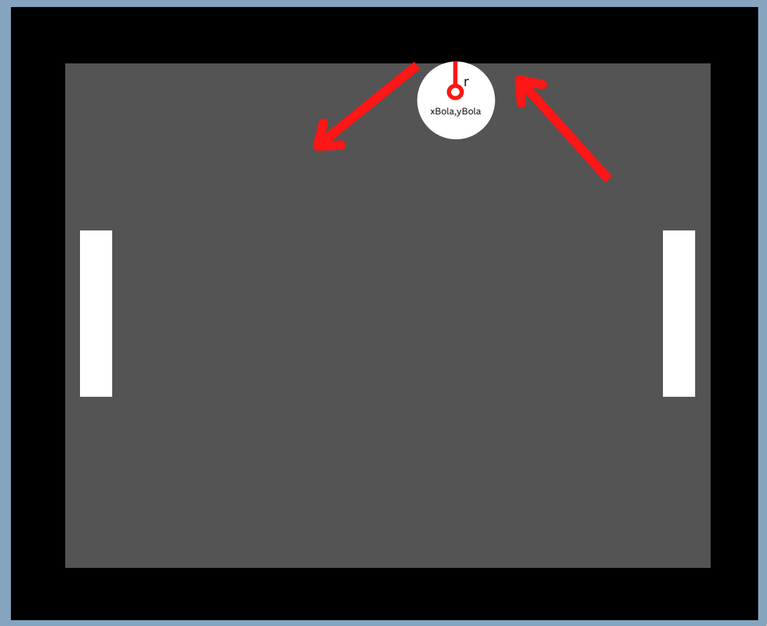
No caso da colisão de cima, o H = 0, então basta pegar o yBola diminuir do raio e verificar se é igual ou menor que 0.

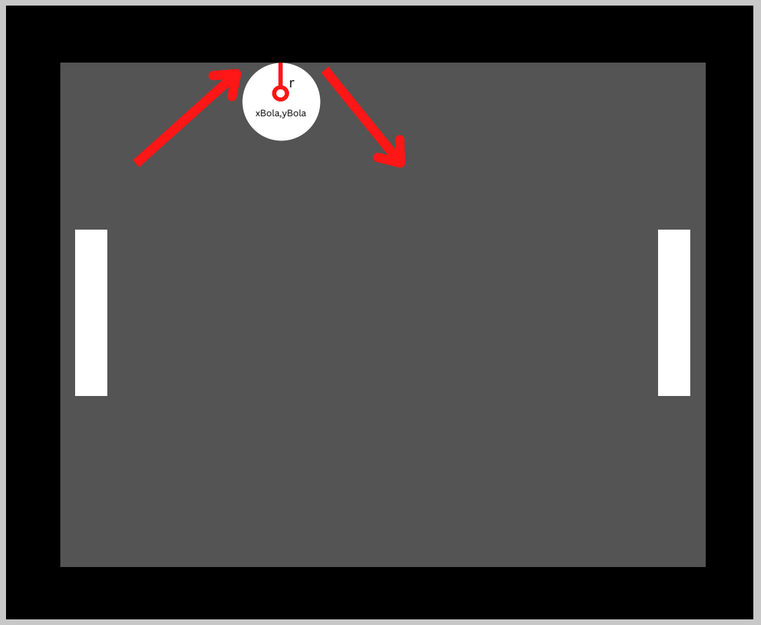
Se for, colidiu.

Bolinha-Parede de Cima

O calculo para ela ricochetear é igualar a velocidade ySpeed a um valor absoluto positivo, pois a velocidade no Y será aumentada positivamente.
Então será dada assim:

```
if(yBola<=0){
ySpeed=abs(ySpeed);
}
```



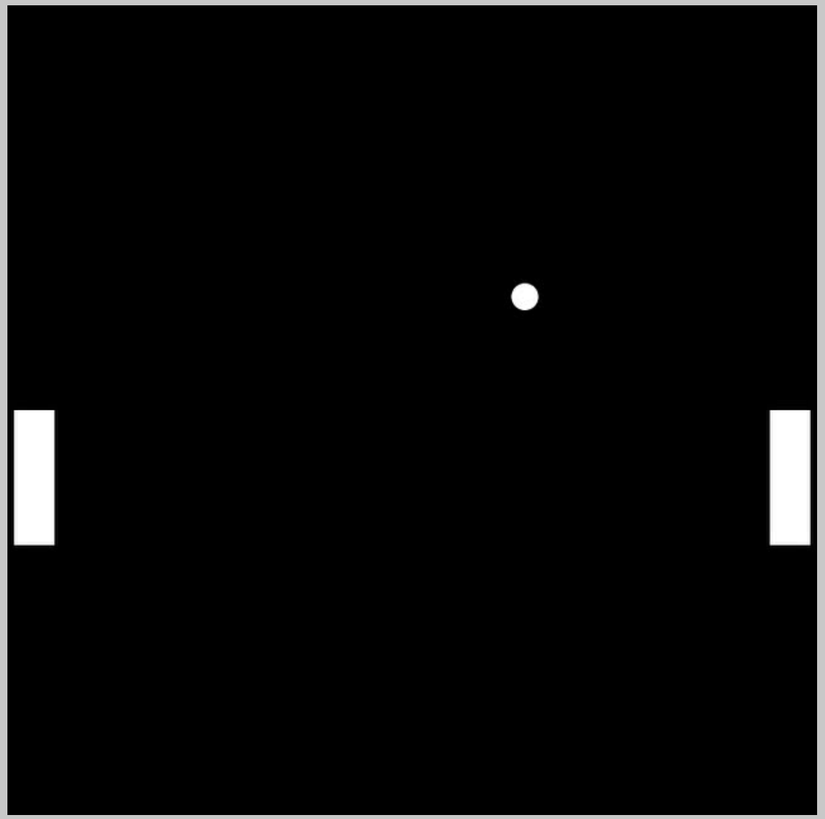


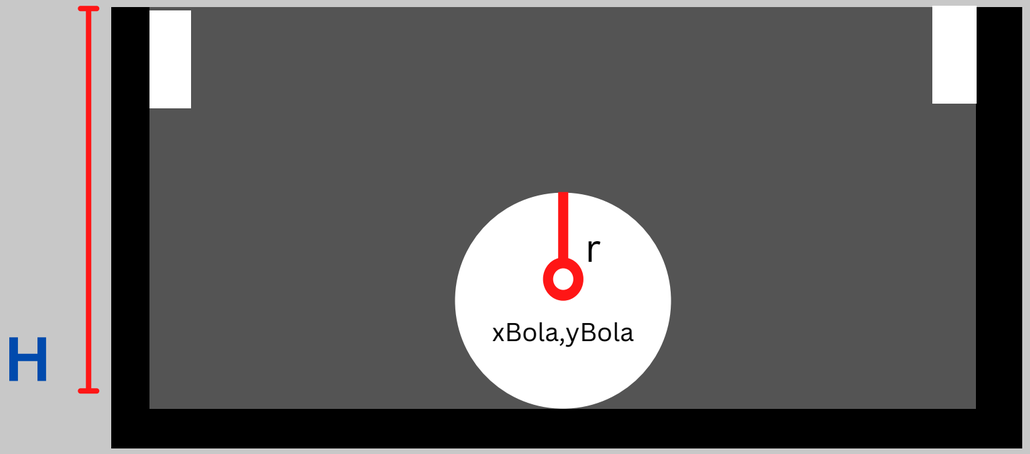
O próximo movimento da bolinha após a colisão será feito com base no sentido da velocidade horizontal.
Se o xSpeed estiver aumentando com valor positivo, ela irá ir para a direita.

Agora, se o xSpeed estiver diminuindo, ela irá para a esquerda

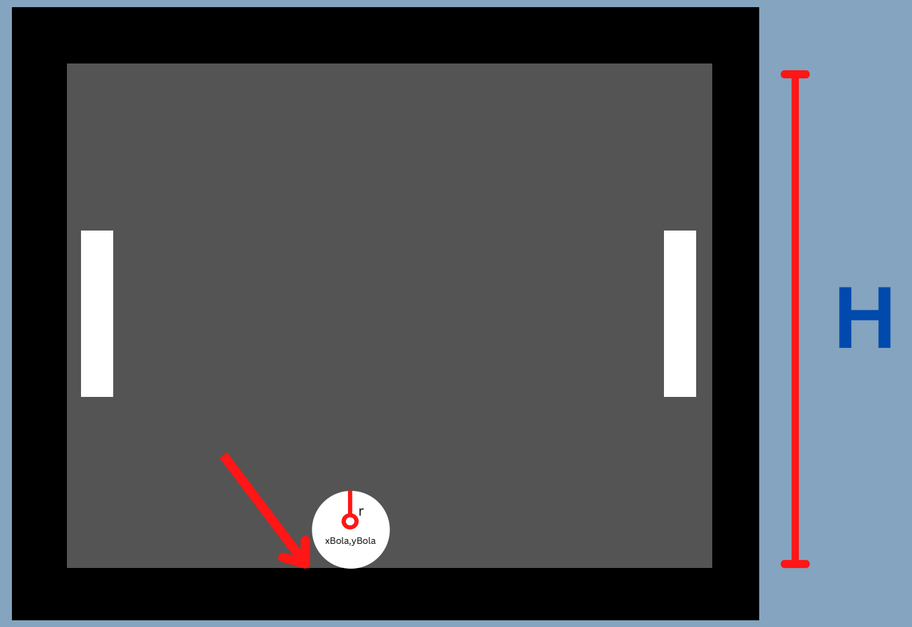
Bolinha-Parede de Cima

O resultado esperado é esse do vídeo ao lado:





Bolinha-Parede de Baixo



Agora vamos testar se ela bateu embaixo.

No caso da colisão de baixo, o H vai ser o seu valor total, então basta pegar o yBola somar com o raio e verificar se é maior ou igual que H.

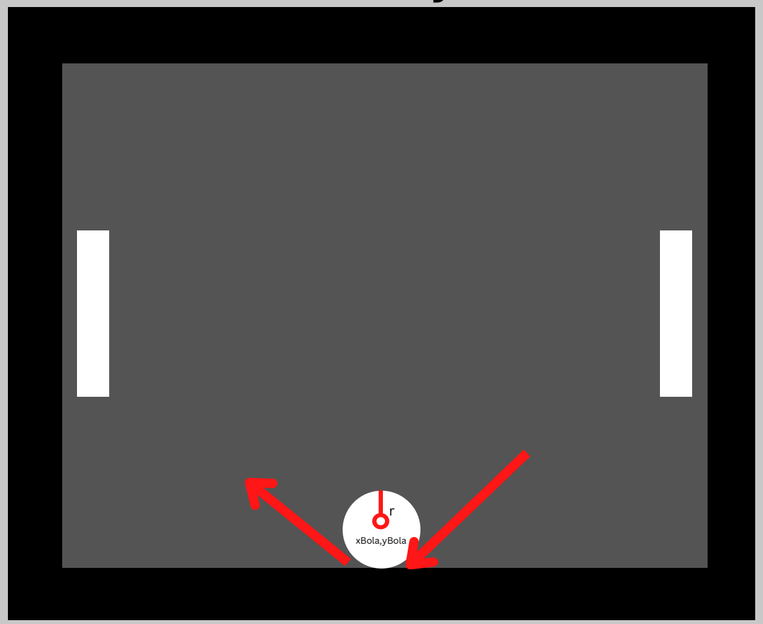
Se for, colidiu.

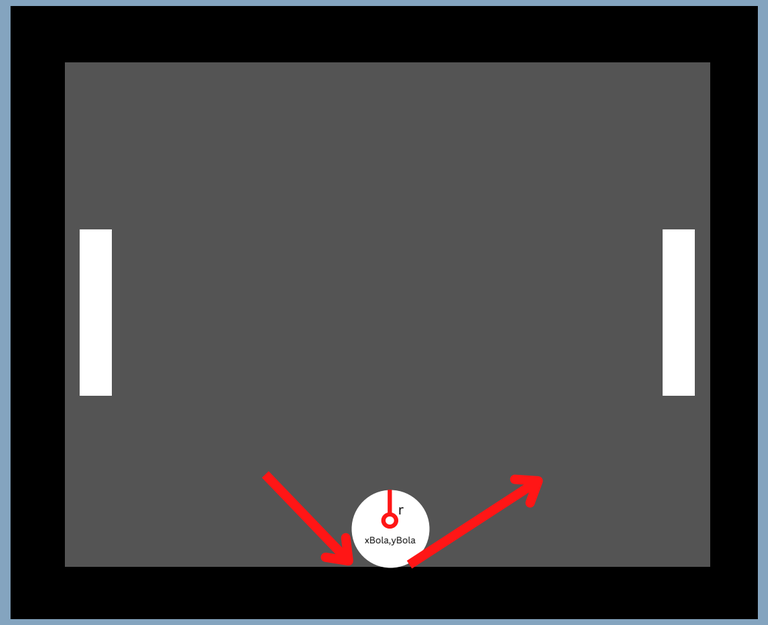
Bolinha-Parede de Baixo

O calculo para ela ricochetear não é muito diferente do outro. A diferença vai ser só igualar a velocidade ySpeed a um valor absoluto negativo , pois a velocidade no Y será aumentada negativamente.

Então será dada assim:

```
if(yBola>=H){
ySpeed=-abs(ySpeed);
}
```





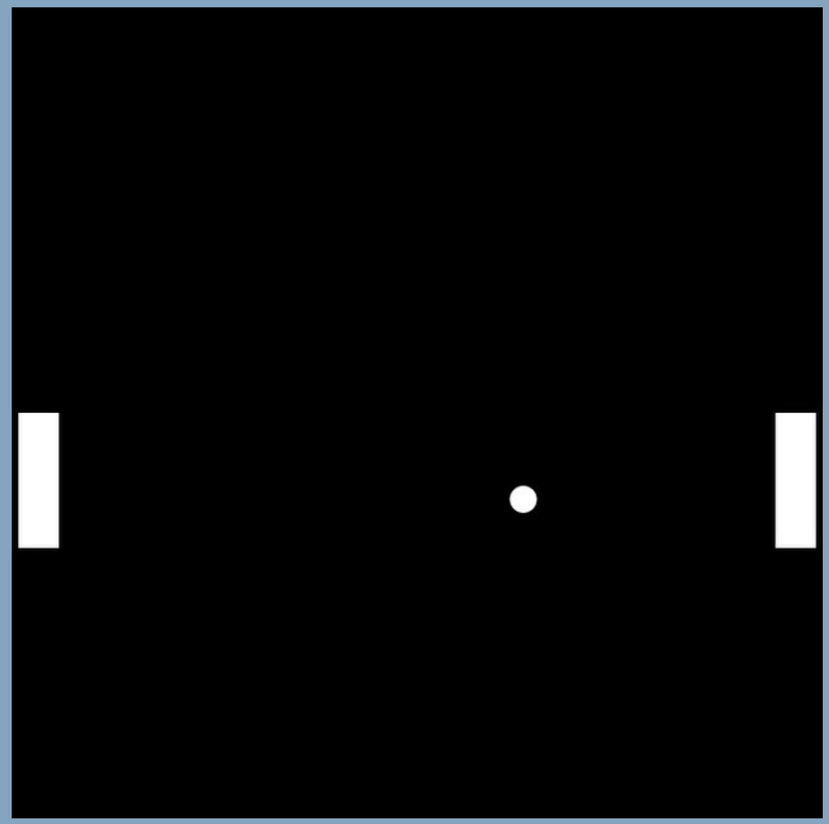
O movimento da bolinha após a colisão será igual ao da colisão de cima. Será feito com base no sentido da velocidade horizontal.

Se o xSpeed estiver aumentando com valor positivo, ela irá ir para a direita.

Agora, se o xSpeed estiver diminuindo, ela irá para a esquerda

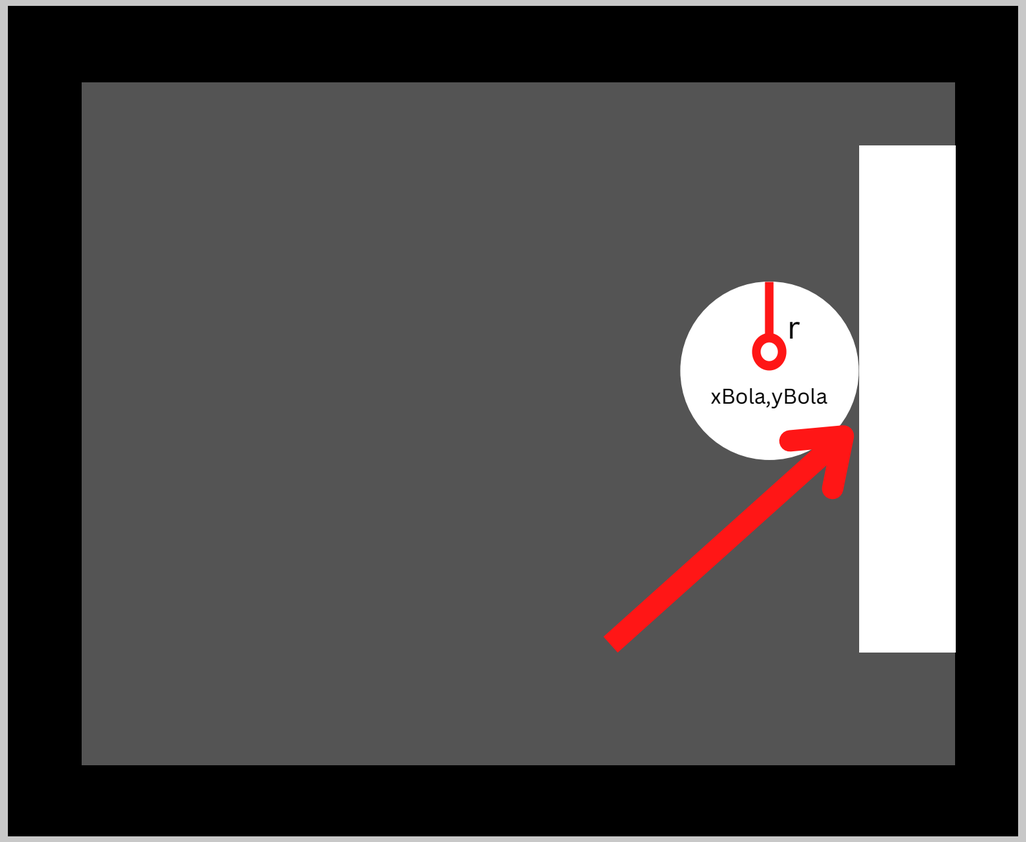
Bolinha-Parede de Baixo

O resultado esperado é esse do vídeo ao lado:

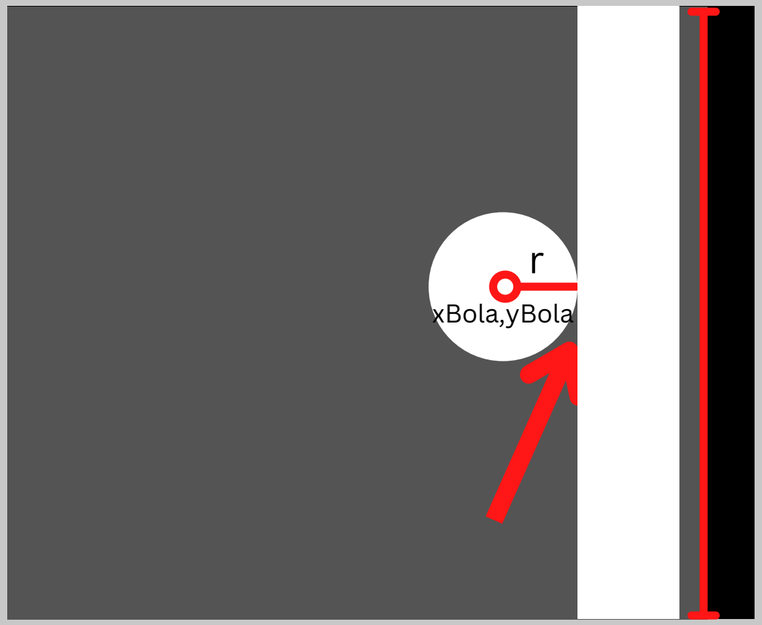


Vamos analisar agora as diferentes colisões entre a bolinha - bastão

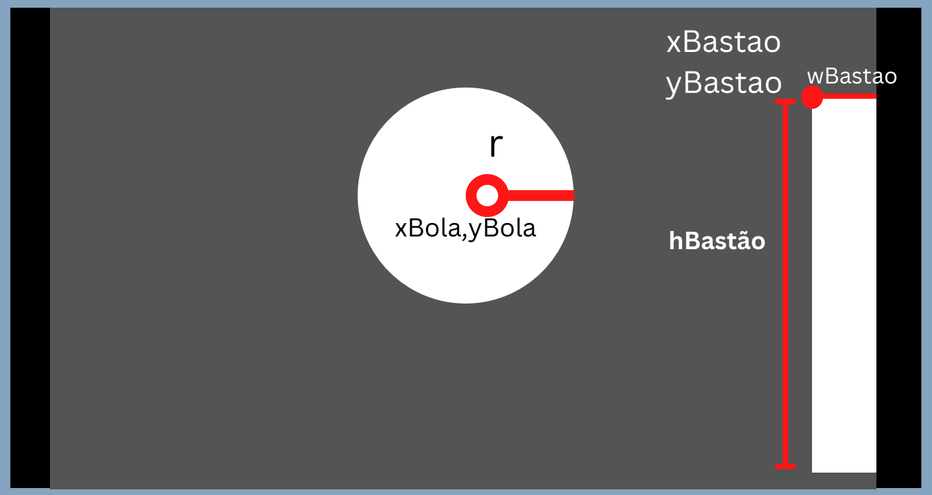
Bolinha-Parede
Tipo de colisão RECT-RECT



Bolinha-Bastão



hBastão

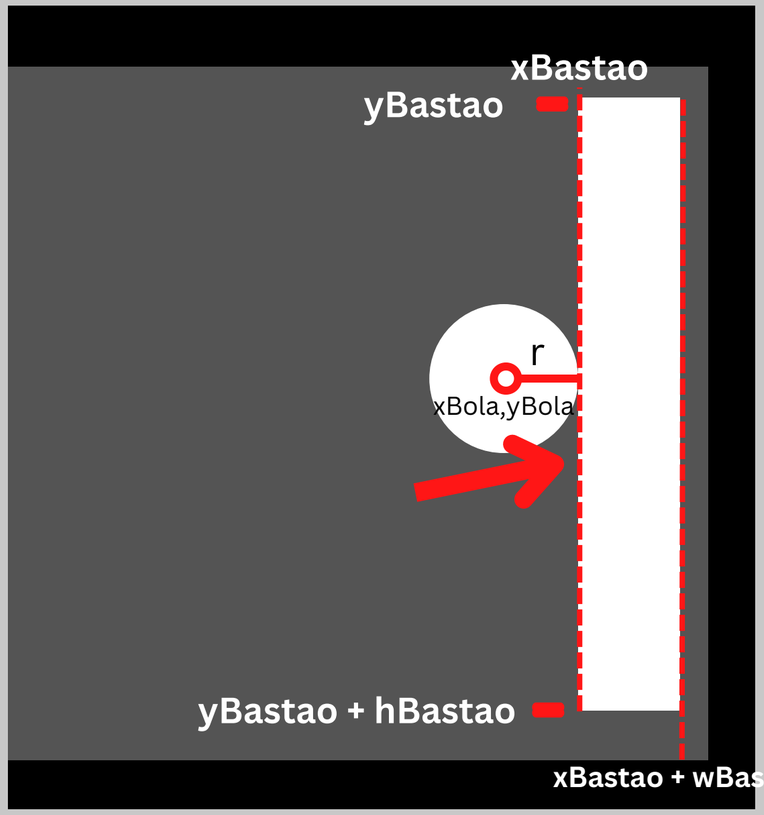


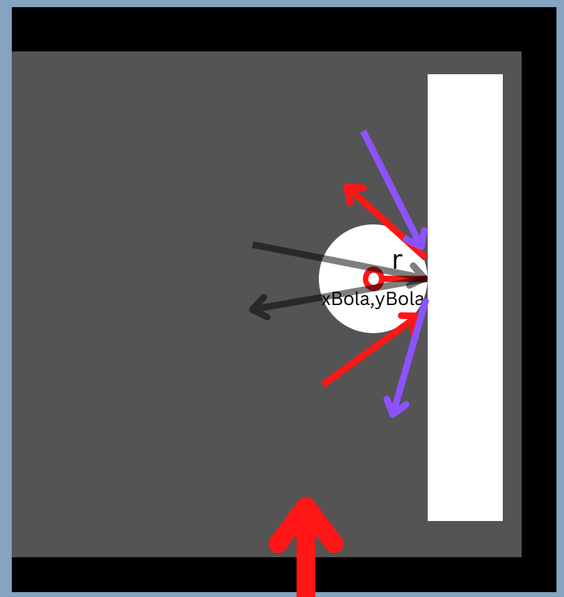
Para testarmos se ela bateu no bastão, precisamos lembrar dos valores referenciais dele, tais como: xBastao, yBastao, wBastao e hBastao.

No caso da colisão frontal, temos que limitar a mesma tanto na largura do bastão como na altura, e depois disso verificar se o xBola é maior ou igual ao xBastao.

Se tudo isso for verdade, houve colisão.

Bolinha-Bastão





Exemplo:

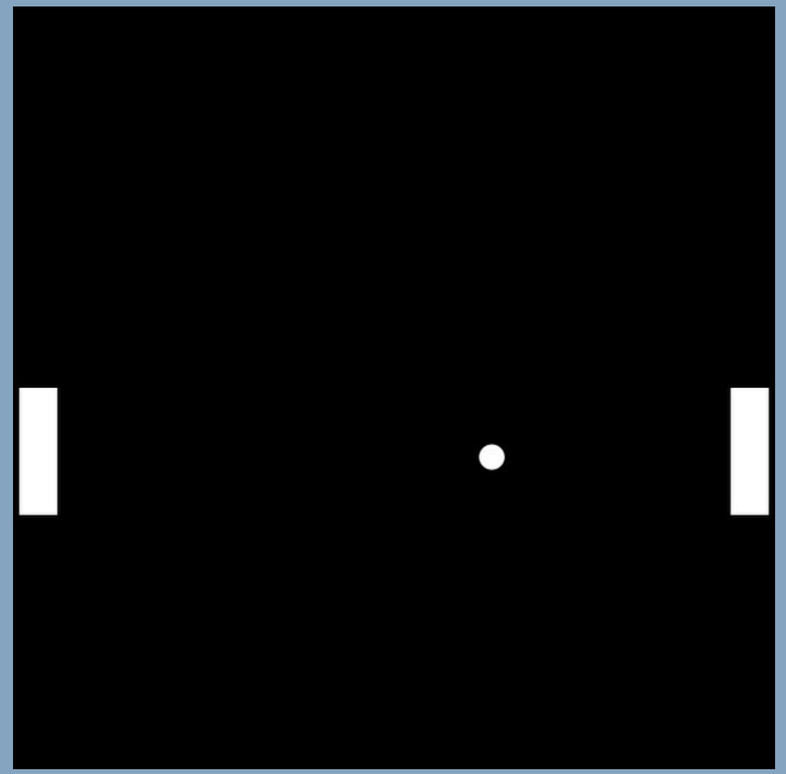
O modelo computacional não é muito dificil.
Ficará assim:

```
if (xBola+r>=xBastao && xBola-r<=xBastao){
if(  yBola-r>=yBastao && yBola+r<=yBastao+hR) {
        xSpeed=-abs(xSpeed);
        }
```

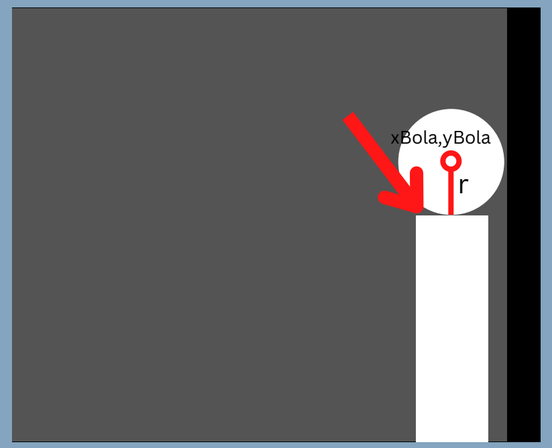
A mudança de sentido só se dará no eixo X, podendo assim ter deslocamentos em vários ângulos

Bolinha-Bastão

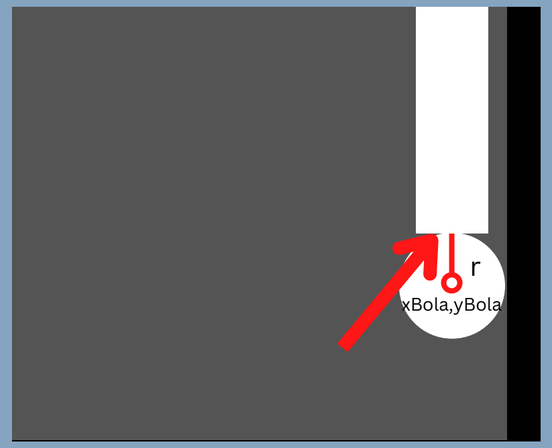
O resultado esperado é esse do vídeo ao lado:



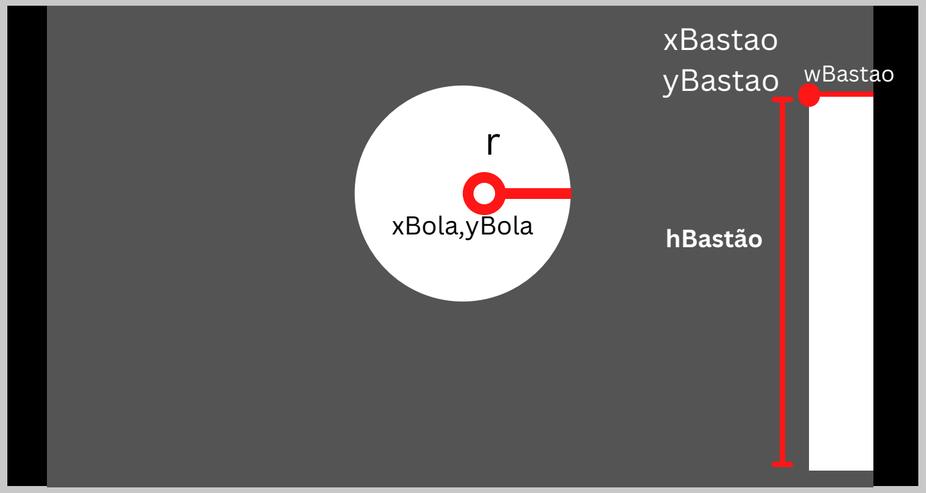
Bolinha-Teto-Bastão
Bolinha-Base-Bastão



Bolinha-Teto-Bastão



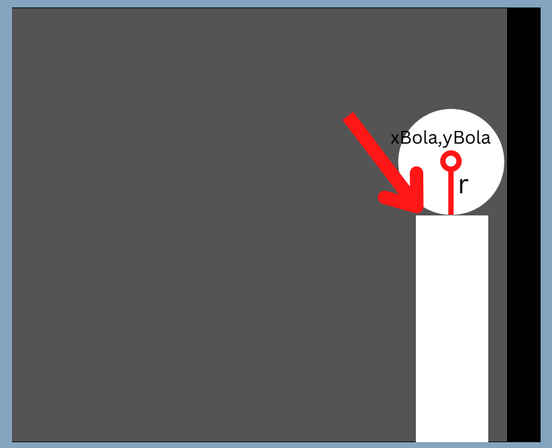
Bolinha-Base-Bastão



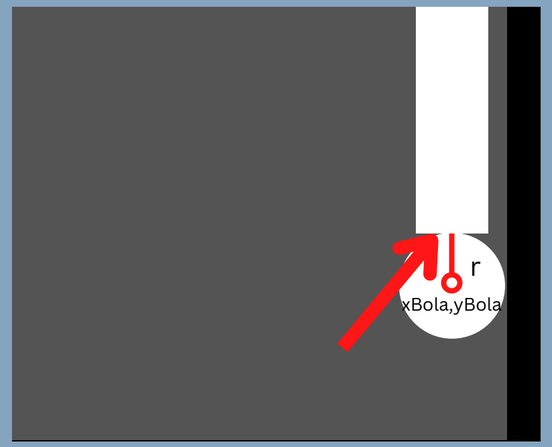
Agora testaremos se ela bateu no teto ou na base do bastão

No caso dessa colisão , temos que limitar a mesma tanto na largura do bastão como na altura, e depois disso verificar.
Se o yBola+r é maior ou igual ao yBastao (para o Teto), ou se o yBola-r é menor ou igual ao yBastao+hBastao (para o Base).
Se tudo isso for verdade, houve colisão.

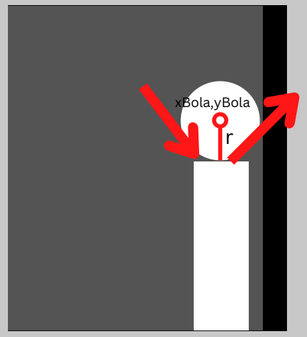
Bolinha-Teto-Bastão
Bolinha-Base-Bastão

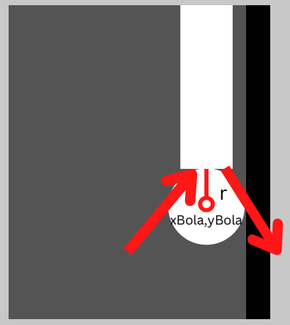


Bolinha-Teto-Bastão



Bolinha-Base-Bastão





Ao contrário da colisão anterior, quando colidir no teto ou na base só alterará o eixo Y da bolinha que será lançada para fora

O modelo computacional seria :

TETO:

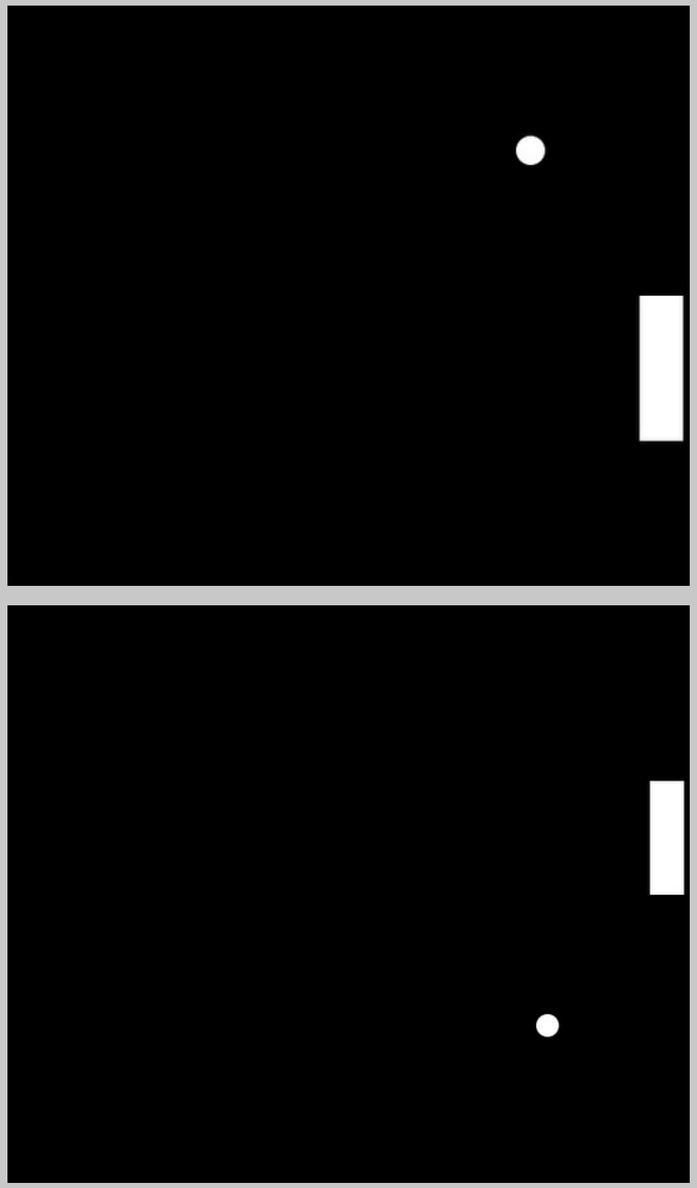
```
 if (xBola-r>=xBastao && xBola+r<=xBastao+wR){
    if (yBola+r>=yBastao && yBola-r<=yBastao) {
   ySpeed=-abs(ySpeed);
}}
```

BASE:

```
 if (xBola-r>=xBastao && xBola+r<=xBastao+wR){
if( yBola-r<=yBastao+hR && yBola+r>=yBastao+hR ) {
   ySpeed=abs(ySpeed);
}}
```

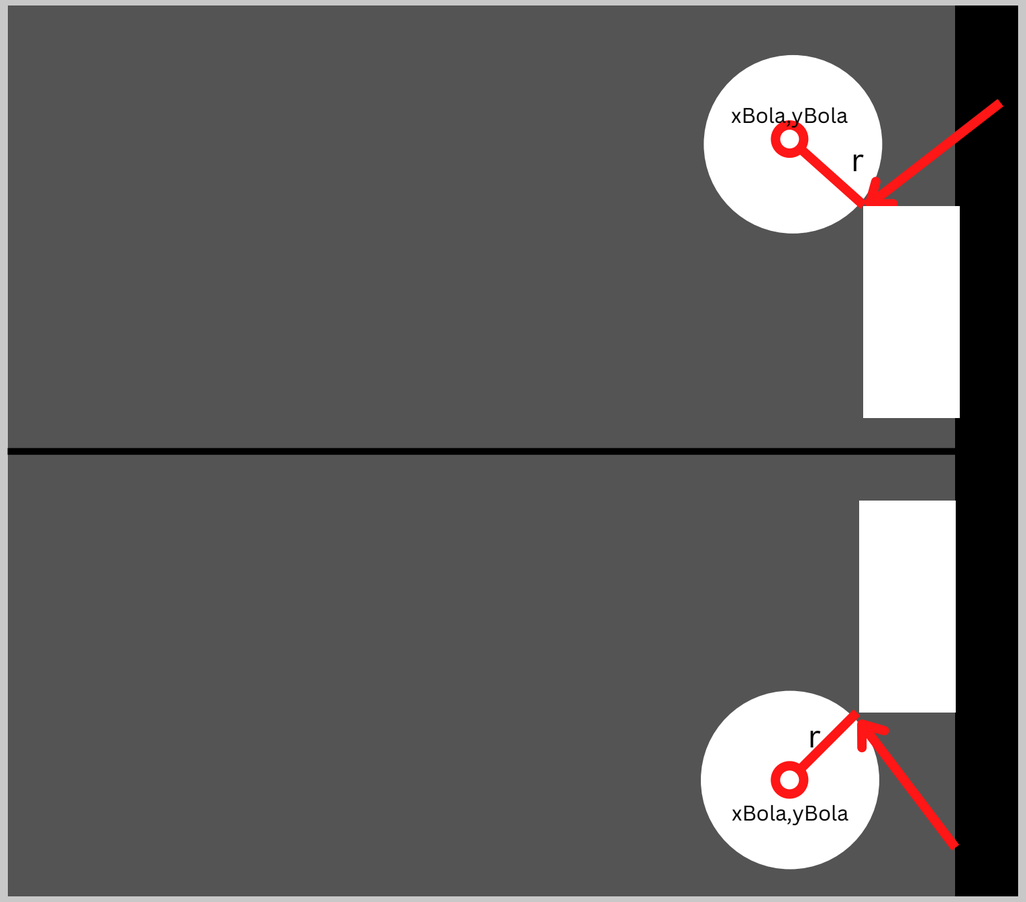
Bolinha-Teto-Bastão
Bolinha-Base-Bastão

O resultado esperado é esse do vídeo ao lado:

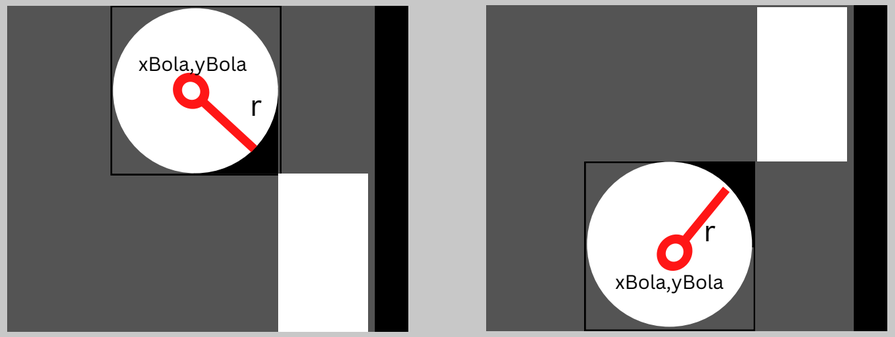


BOLINHA-BASTÃO

Bolinha-Quina-Parede
Tipo de colisão PONTO-CIRCULO



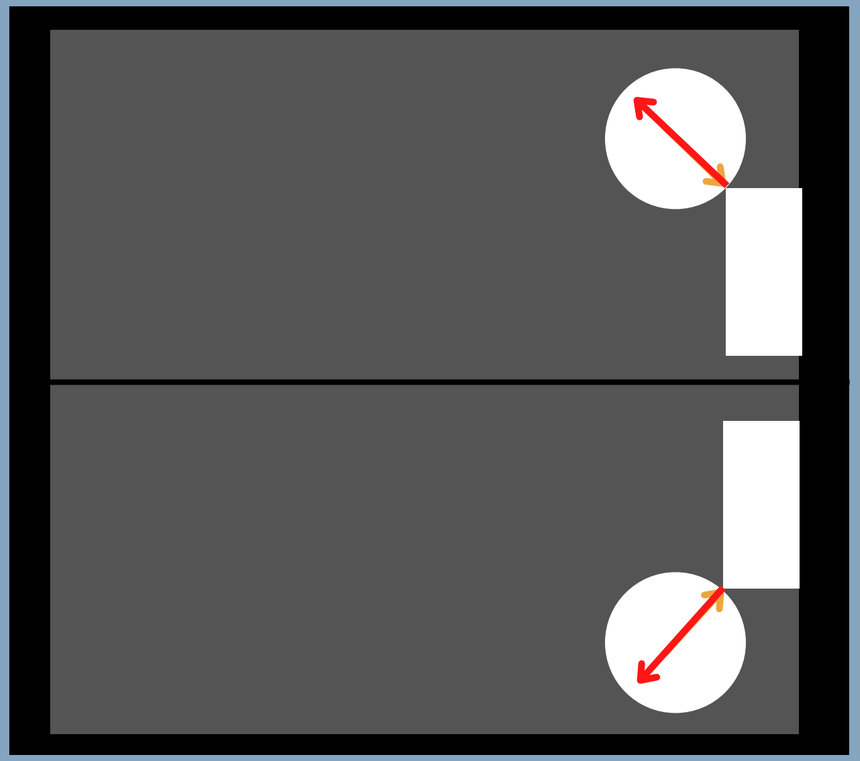
Essa é uma colisão bem específica, pois é PONTO-CIRCULO. E a explicação para isso está na figura abaixo.



Se utilizássemos RECT-RECT, a colisão iria acontecer antes de chegar ao Raio do círculo.
Então para fazermos essa colisão, faremos PONTO-CÍRCULO, usando a função DIST(); visto em aula.

Pegaremos a distancia entre o centro do círculo e as quinas e verificaremos se é menor ou igual a raio do círculo.
Se for, houve colisão

O ricochete também é específico, pois vai inverter tanto o eixo X como o eixo Y



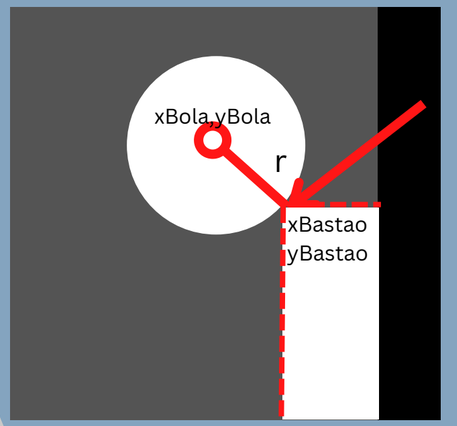
O modelo computacional seria :

Bolinha-Quina-Cima:

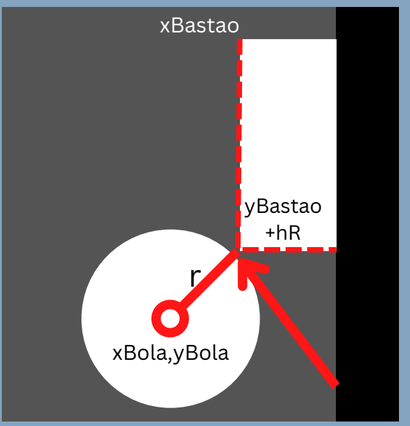
```
int quinaCima=dist(xBola, yBola, xBastao, yBastao);
 if (quinaCima<=r) {
    xSpeed=-abs(xSpeed);
    ySpeed=-abs(ySpeed);
}
```

Bolinha-Quina-Baixo:

```
int quinaBaixo= dist(xBola, yBola, xBastao, yBastao+hR);
if (quinaCima<=r) {
   xSpeed=-abs(xSpeed);
   ySpeed=abs(ySpeed);
}
```

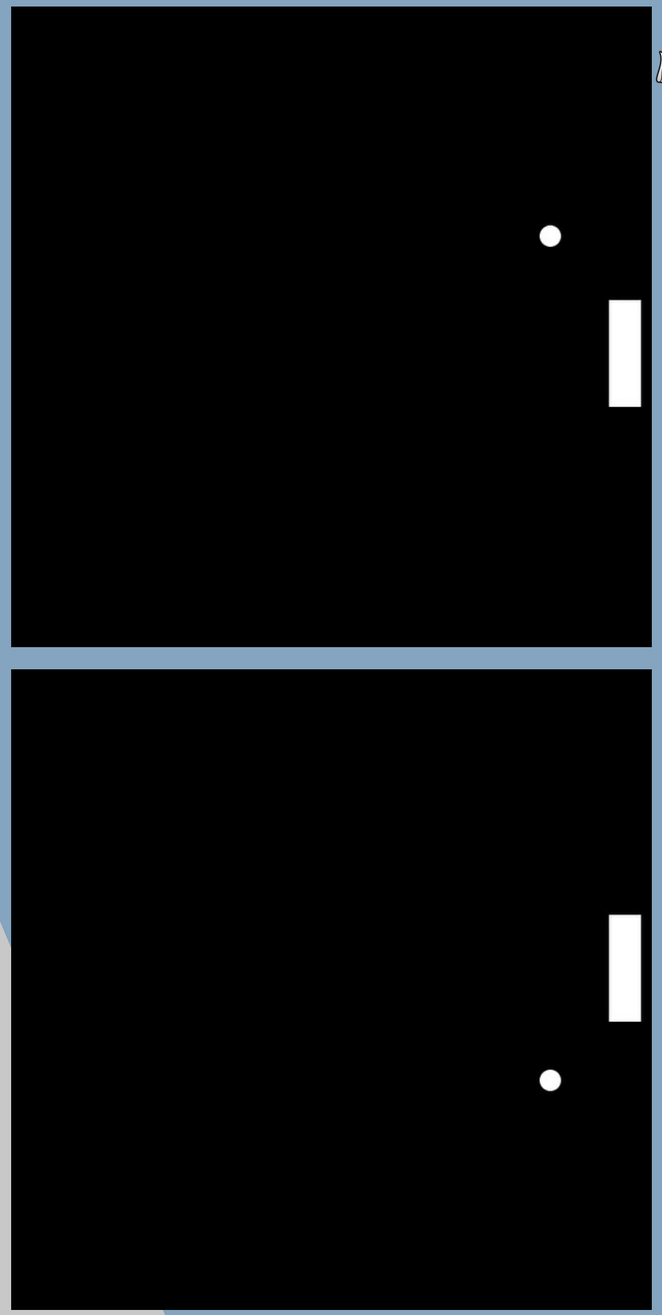


Bolinha-Quina-Cima



Bolinha-Quina-Baixo

O resultado esperado é esse do vídeo ao lado:



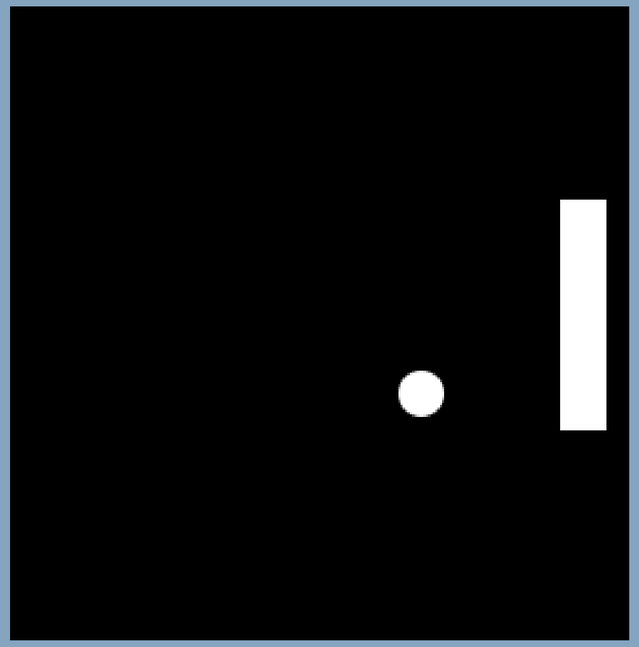
O código para o Jogo
Pong com Colisões
está na pasta

FIM

Matemática Aplicada à Multimídia I

PONG

Teste de Colisão



Daniel Cardeal - 541875

Profª - Mara Bonates