

Tutorial: Programação em arduino construindo um videogame - Pong

Hugo Naoki Sonoda Okumura Pedro Chouery Grigolli

December 8, 2025

VERSÃO DO PROFESSOR - Não deve ser divulgada

Introdução

Neste tutorial vamos introduzir a programação em Arduino através da montagem e implementação de um vídeogame PONG com a opção de 1 ou 2 jogadores, com saída de imagem para uma Televisão.

Nesse material, será trabalhado:

- Uma apresentação do microcontrolador Arduino
- Apresentação do jogo PONG
- Uma introdução inicial para a programação no microcontrolador
- Montagem e implementação do vídeogame

Lista de Materiais

- Potenciômetros 10KΩ
- Capacitores 50V 10 μ F
- Push Buttons
- Cabo RCA Y 1 Fêmea 2 Macho
- Resistores 1KΩ e 470R
- Televisão com entrada RCA

Sumário

1	Conceitos Iniciais	2
1.1	Sobre o Arduino UNO	2
1.1.1	Componentes Internos e Limitações de Hardware	2
1.1.2	Conexão Com Componentes Externos	3
1.2	Programação Básica Em Arduino	4
1.2.1	Programação Embarcada	4
1.2.2	Diferenças com Programação Convencional	4
1.2.3	Lidando com Entrada e Saída	4
1.3	Arduíno IDE	5
1.4	Compilação e Carregamento de código no Arduino IDE	5
1.4.1	Configuração Inicial	5
1.4.2	Compilação (Verificar)	6
1.4.3	Carregamento (Upload)	7
2	Desenvolvimento do Game	8
2.1	Regras do PONG	8
2.2	Circuito montado	9
2.3	Estrutura Geral e Funcionamento do Loop de Jogo	11
2.4	Controle dos Jogadores Utilizando Potenciômetros	12
2.5	Lógica da Bola e Mecânica de Colisões	12
2.6	Pontuação e Transição de Estados	14
3	Avaliação	15

1 Conceitos Iniciais

1.1 Sobre o Arduino UNO

O Arduino é uma plataforma de hardware e software open-source baseada em uma placa de microcontrolador (PCB) que simplifica a criação de projetos eletrônicos interativos.

1.1.1 Componentes Internos e Limitações de Hardware

A placa Arduino, como a UNO, é essencialmente um PCB (Placa de Circuito Impresso) que contém um Hardware principal, o microcontrolador (geralmente um ATmega328P). Este microcontrolador e seus componentes auxiliares são os Componentes Internos do Arduino.

O microcontrolador é um sistema de computador em miniatura, mas possui Limitações de Hardware significativas em termos de capacidade de processamento, memória RAM e armazenamento.

- **Apesar das Limitações:** A limitação de recursos não é uma desvantagem para sistemas dedicados (embarcados). Pelo contrário, ela garante baixo consumo de energia, custo reduzido e controle direto sobre os periféricos.
- **O Desafio da Programação:** Programar o Arduino exige otimização do código para respeitar a memória e o poder de processamento limitados

1.1.2 Conexão Com Componentes Externos

O Arduino conecta-se a Componentes Externos por meio de seus Pinos. A interface entre o microcontrolador e o mundo físico é a Eletrônica, que pode envolver o uso de resistores, capacitores e outros componentes passivos e ativos.

- **Pinos:** Os Pinos na placa (PCB) são a interface de comunicação. Eles são identificados por número e podem ser configurados via software para atuar como Entrada ou Saída.
- **Leitura de Datasheet:** Para conectar componentes externos de forma segura e eficiente, é fundamental consultar o Datasheet (Leitura de Datasheet). Isso garante que os limites de tensão e corrente do Arduino e do componente externo sejam respeitados, evitando danos.
- **Entrada e Saída:**
 - **Entrada:** Permite ao Arduino receber dados, como a variação de um potenciômetro (usando Leitura Analógica) ou o acionamento de um botão (Leitura Digital).
 - **Saída:** Permite ao Arduino controlar componentes, como ligar um LED (Transmissão Digital) ou enviar o sinal de vídeo para a TV (Transmissão Digital/Analógica).

Compreendendo o hardware do Arduino, seus componentes internos e como seus Pinos atuam como interface de Entrada e Saída, o próximo passo é aprender a linguagem e o ambiente que nos permite controlar esses recursos. É aqui que entra a Programação Básica em Arduino e as particularidades da Programação Embarcada.

Características	Programação Convencional (PC)	Programação Embarcada (Arduino)
Execução	Orientada a eventos ou processos sequenciais (e.g., abrir um app, fechá-lo)	Focada em um loop contínuo e tempo real (monitorar entradas e controlar saídas)
Ambiente	Sistema Operacional completo (Windows, Linux, macOS)	Executa diretamente no <i>hardware</i> (bare-metal ou firmware), sem um SO completo
Interação	Principalmente via tela, teclado, mouse	Principalmente via pinos de Entrada/Saída para interagir com o mundo físico

Table 1: Comparação entre programação convencional e programação embarcada.

1.2 Programação Básica Em Arduino

1.2.1 Programação Embarcada

A programação embarcada é o desenvolvimento de software que é executado em um Hardware dedicado, chamado de microcontrolador, geralmente sendo restrito a alguns recursos, como Kilobytes de RAM ou armazenamento e velocidades de processamento da CPU mais lentas, o que normalmente não acontece na programação convencional, visto que tanto o armazenamento quanto a velocidade da CPU é muito mais rápida e robusta.

1.2.2 Diferenças com Programação Convencional

A programação convencional, em um PC, e o desenvolvimento em Arduíno são diferentes em vários aspectos. Aqui será detalhado os principais aspectos que diferem o desenvolvimento embarcado para a programação convencional. Observe a tabela 1

1.2.3 Lidando com Entrada e Saída

Na programação no Arduíno, tudo gira em torno de entrada e saída, portanto, é de suma importância dominar essa parte.

O *sketch* (estrutura de código de um programa Arduíno) possui dois métodos principais:

- **setup()**: Executada uma única vez ao ligar ou resetar o Arduíno. É usada para inicializar pinos (definir se são de entrada ou saída) e configurar bibliotecas ou periféricos.
- **loop()**: Executada continuamente após setup() ter sido executada. É nessa função que o código principal do videogame residirá.

1.3 Arduíno IDE

Outra grande diferença da programação embarcada com Arduíno é a necessidade de uma IDE (*Integrated development environment*) própria para escrita de código, compilação e upload para dentro do microcontrolador.

1.4 Compilação e Carregamento de código no Arduino IDE

Compilar e carregar seu código são etapas cruciais que transformam seu sketch em um programa executável no microcontrolador.

1.4.1 Configuração Inicial

Antes de compilar, você deve garantir que o Arduino IDE esteja configurado corretamente:

- **Placa**: Vá em **Ferramentas ↗ Placa** e selecione **”Arduino Uno”**.
- **Porta**: Conecte o Arduino ao computador via USB. Vá em **Ferramentas ↗ Porta** e selecione a porta serial correta (geralmente identificada com o nome da placa, ex: COM3 ou /dev/ttyACM0).

Figure 1: Seleção da Placa

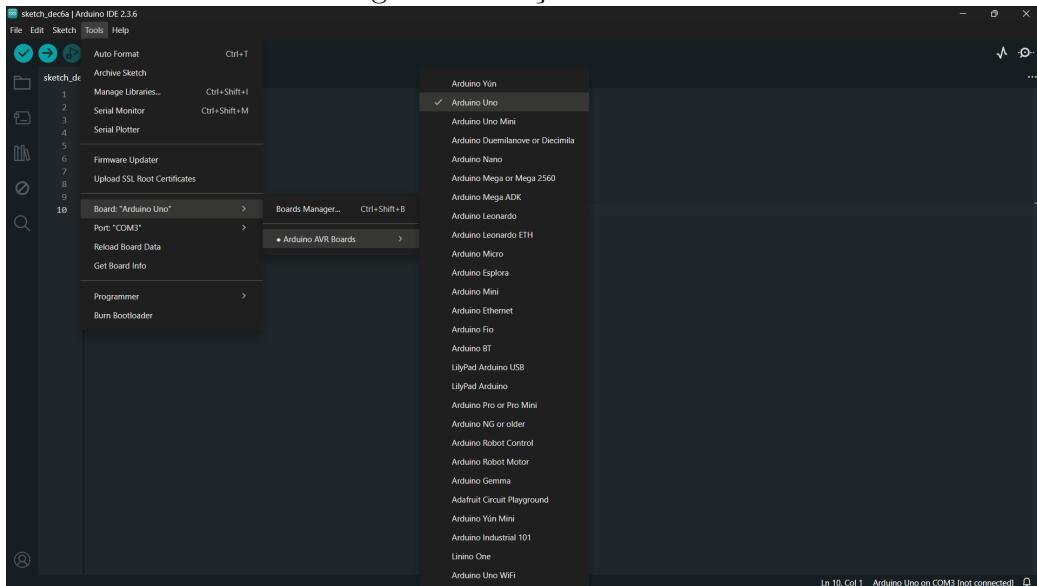
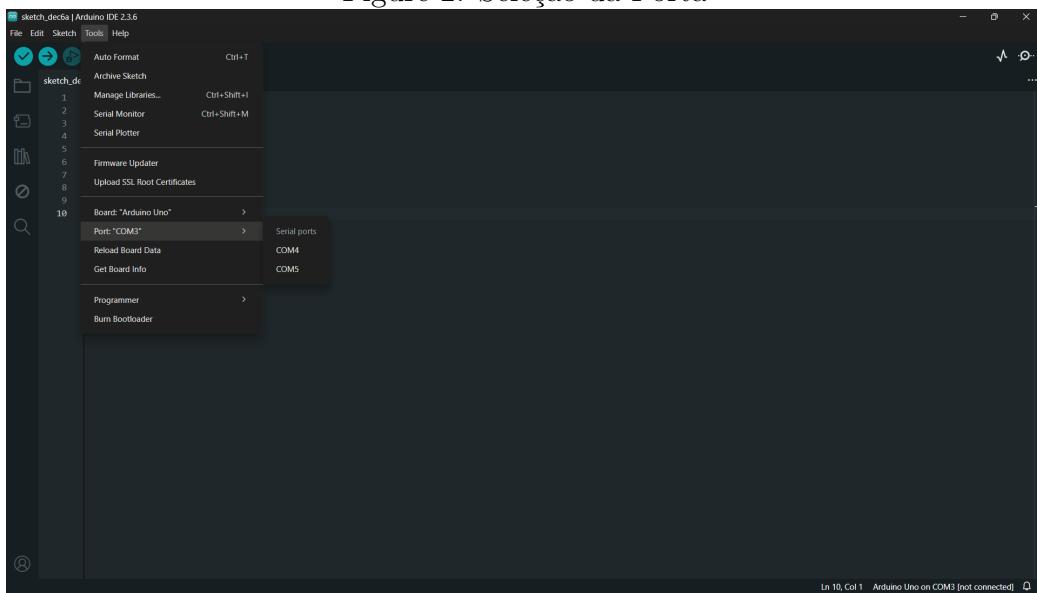


Figure 2: Seleção da Porta

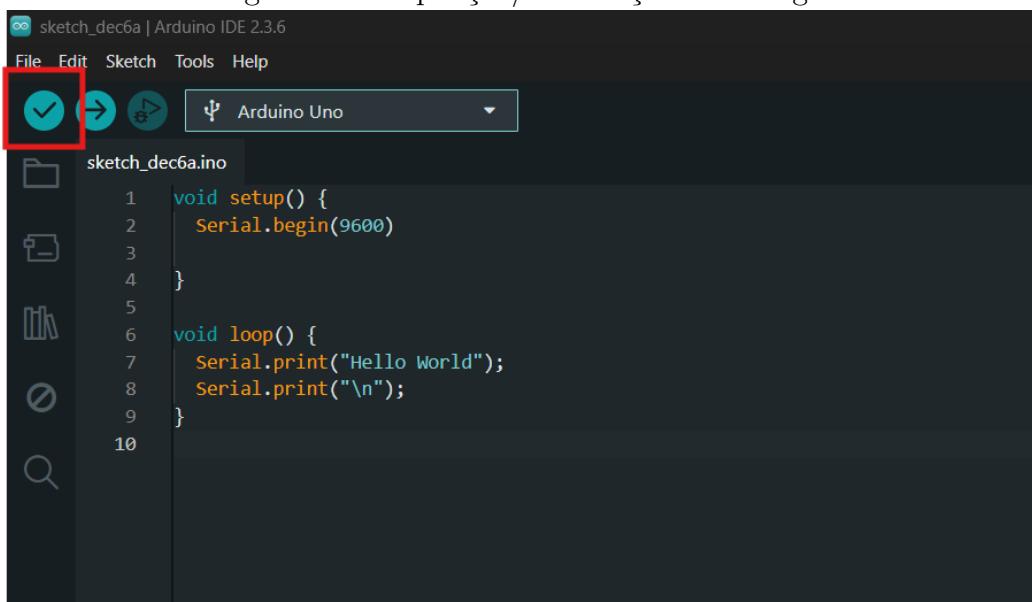


1.4.2 Compilação (Verificar)

A compilação é o processo de traduzir o código escrito (linguagem humana) para o código de máquina binário que o ATmega328P consegue executar.

- Ação: Clique no ícone de ”Verificar” (ou Check Mark / Visto) na barra de ferramentas.
- Resultado: O IDE tentará compilar seu código. Se houver erros de sintaxe, o processo falhará e a área de console (parte inferior do IDE) exibirá mensagens vermelhas indicando a linha e o tipo de erro. Se for bem-sucedido, o IDE informará a quantidade de memória utilizada pelo programa.

Figure 3: Compilação/Verificação do Código



1.4.3 Carregamento (Upload)

O carregamento é o processo de gravar o código compilado (arquivo .hex) na memória flash do microcontrolador.

- Ação: Clique no ícone de ”Carregar” (ou Seta para a Direita) na barra de ferramentas.
- O IDE primeiro compila o código (se ainda não o fez) e, em seguida, usa o bootloader da placa para transferir os dados via USB.
- As luzes RX e TX do Arduino piscarão rapidamente durante a transferência. Uma mensagem de ”Carregamento concluído” será exibida no

console. Após o carregamento, o programa começa a ser executado automaticamente, iniciando com a função `setup()` e, em seguida, entrando no `loop()`.

Figure 4: Compilação e Upload do Código



2 Desenvolvimento do Game

Nessa seção será focado na implementação do PONG. Não será mostrado neste documento o código pronto, ao invés disso, será apresentado a lógica por trás de cada componente do jogo, para que você use sua criatividade de como fazer a implementação do jogo. Recomenda-se a leitura da biblioteca TVout (disponível no Github), para que facilite a sua solução.

Observação: No .zip da biblioteca TVout, existe uma outra biblioteca chamada "TVoutfonts", que você deve extraí-la e colocar na pasta *libraries* junto com a própria TVout. Essa biblioteca faz o controle do tamanho do texto dos *prints* da TVout, o que será muito útil para a implementação do jogo.

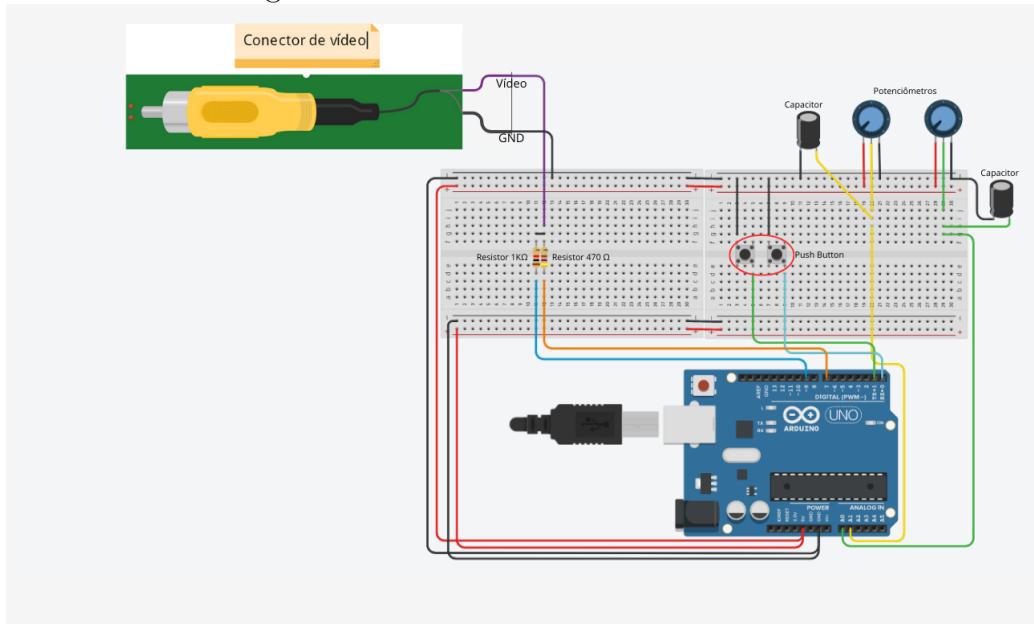
2.1 Regras do PONG

O PONG, desenvolvido pela Atari e lançado em 1972, é considerado um dos primeiros videogames da história e com certeza um dos mais famosos. Suas regras são:

- Cada jogador controla um raquete vertical e tentar acertar a bola para que ela vá em direção ao campo do oponente
 - O jogador marca um ponto quando a bola ultrapassa a raquete do oponente e sai dos limites do campo lateralmente.
 - A raquete do jogador se move verticalmente, neste caso controlada por um potenciômetro.

2.2 Circuito montado

Figure 5: Circuito montado no TinkerCad



Para este projeto foi utilizado os seguintes componentes:

- **2x Potenciômetros 10KΩ:** Serão utilizados como controles para os jogadores, onde a leitura analógica definirá a posição do jogador na tela.
 - **2x Capacitores 50V 10μF:** Conectados junto com os potenciômetros para a estabilização da leitura, oferecendo um controle mais preciso do movimento.
 - **2x Push Buttons:** Utilizado para o controle do menu inicial do jogo, não é obrigatório ter 2, apenas 1 será necessário, porém deverá ter uma lógica diferente do que é feito neste artigo.

- **1x RCA Y 1 Fêmea 2 Macho:** Será utilizado para a transmissão de imagem para a televisão. Para este projeto, será utilizado apenas a saída de vídeo. Recomendado que utilize algum cabo que não esteja sendo usado, pois, para a montagem do circuito, será desencapado o cabo de saída de vídeo para a conexão do circuito.
- **1x Resistor de 1KΩ e 1x Resistor de 470R:** Estes resistores estarão conectados em série com o fio de vídeo do cabo RCA. O resistor de 470R será conectado ao pino 7 do Arduino, que é o pino que a biblioteca TVout utiliza para o vídeo, e o resistor de 1KΩ no pino 9, que é utilizado para sincronização pela biblioteca (Figura 2).

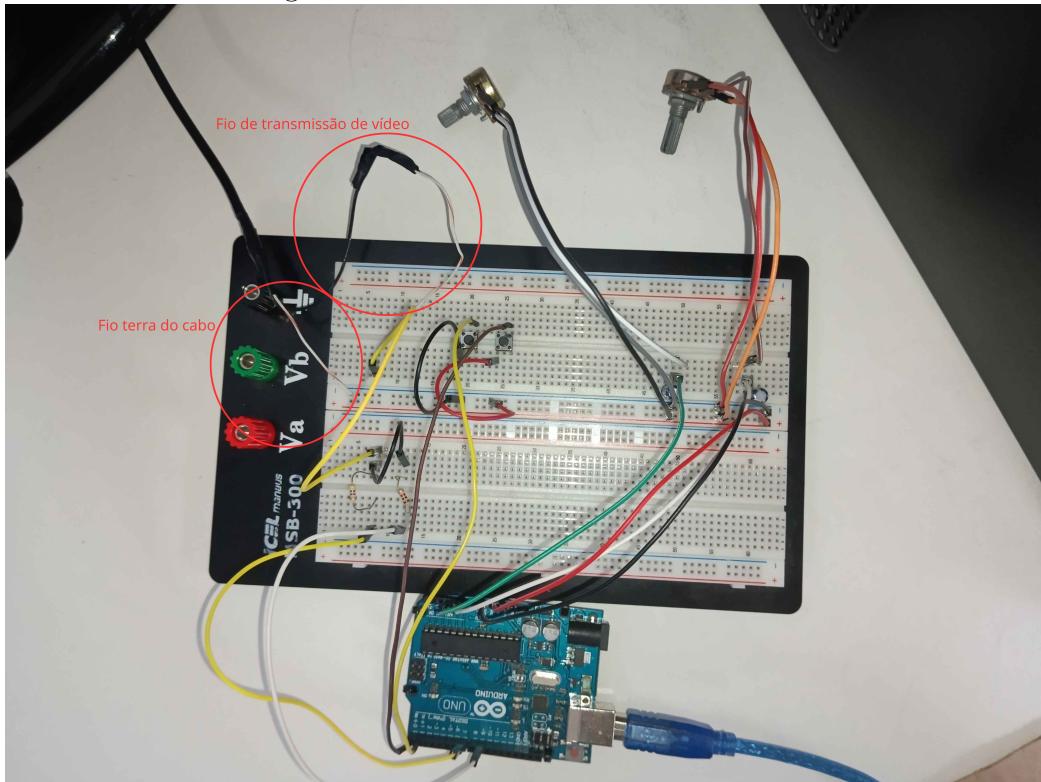
Figure 6: Tabela de pinos que o TVout utiliza para cada microcontrolador

MCU	SYNC	VIDEO	AUDIO	Arduino	SYNC	VIDEO	AUDIO
m168,m328	B 1	D 7	B 3	NG,Decimila,UNO	9	7	11
m1280,m2560	B 5	A 7	B 4	Mega	11	A7(D29)	10
m644,m1284p	D 5	A 7	D 7	sanguino	13	A7(D24)	8
m32u4	B 5	B 4	B 7	Leonardo	9	8	11
AT90USB1286	B 5	F 7	B 4	--	--	--	--

Fonte: Gihub da Biblioteca (Em Inglês)

Para a conexão do cabo RCA, terá que fazer o desencapamento do fio de vídeo. Fazendo isso, no interior do cabo, haverá dois materiais condutores, um na parte mais interna do cabo e um envolta de uma malha isolante. O fio mais interno é o fio que faz a transmissão de vídeo e esse é o que deverá ser conectado aos resistores. O mais externo é o fio terra e deverá ser conectado ao GND do Arduino. Na (Figura 3) tem uma foto mostrando como ficaria o circuito fisicamente.

Figure 7: Circuito montado fisicamente



2.3 Estrutura Geral e Funcionamento do Loop de Jogo

Com o hardware pronto em mãos, o desenvolvimento agora se concentra muito mais na construção da lógica interna principal, onde o jogo irá funcionar de forma continua até que seja interrompido. Como já foi dito aqui, esse comportamento ocorrerá dentro da função nativa do Arduino `loop()`, que é executado indefinidamente até que seja interrompido por forças maiores. Nesse trecho, devemos desenvolver as seguintes partes:

- Atualiza a posição dos elementos na tela
- Avalia entradas dos jogadores
- Verifica colisões
- Controla pontuação
- Decide transições entre diferentes fases do jogo

(ou seja, basicamente tudo). A cada iteração então, basicamente, o sistema verifica em que estado se encontra (menu, inicio, etc) e executa apenas as funções pertinentes. Essa divisão de responsabilidade entre os estados garante organização lógica, previsibilidade de execução e evita sobrecarga de processamento, o que é crucial trabalhando com Arduino e microcontroladores num geral.

2.4 Controle dos Jogadores Utilizando Potenciômetros

Cada jogador controla um raquete cujo movimento é determinado pela leitura de um potenciômetro. Como esse tipo de entrada produz um valor analógico, o Arduino converte a tensão lida em um valor numérico digitalizado, normalmente entre 0 e 1023. Esse valor, então, deve ser mapeado para a altura útil da tela, fazendo todas as conversões matemáticas necessárias e garantindo que a raquete se move proporcionalmente à rotação do potenciômetro.

```
// Lê valor do potenciômetro
x = analogRead(A0);
// O transforma em um valor entre 20 e 80
x = map(x, 0, 1023, 20, 80);
p1_cima = x + 12;
// Utiliza o valor conseguido com a leitura
// para determinar a posição do jogador
p1_baixo = x - 12;
// Desenha o personagem do jogador
TV.drawLine(5, p1_baixo, 5, p1_cima, WHITE);
```

2.5 Lógica da Bola e Mecânica de Colisões

Tudo gira em torno da bola e do seu posicionamento, portanto o jogo requer tratamento contínuo da sua posição e direção. Para representar seu movimento, o sistema deve manter internamente dois valores de posições e dois de velocidade, que correspondem ao deslocamento horizontal e vertical por ciclo. Dentro do loop, a cada iteração, a bola tem a sua velocidade somada a sua posição atual, fazendo com que ela ande pela tela e após isso, o sistema deve verificar se a bola não está ultrapassando os limites da tela ou colidindo com alguma raquete.

A colisão com as bordas superior e inferior é mais simples de ser resolvida: inverte-se a componente vertical da velocidade, simulando um rebote, por exemplo. Já a colisão com as raquetes é um pouco mais complexa: além de

inverter a componente horizontal da velocidade, ajusta-se na vertical dependendo da região colidida com a raquete. No centro a bola seguirá reto para a direção oposta, nas bordas, a bola se movimentará para mesma direção que a borda colidida, ou seja, se for colidida na parte de baixo, a bola irá para baixo, e na parte de cima, para cima.

```
{
Colisão com o jogador
    dir_x = 1;
    if (circle_y <= p1_cima && circle_y > p1_baixo + 16)// bateu em cima
    {
        dir_y = 1;
    }
    if (circle_y <= p1_cima - 8 && circle_y > p1_baixo + 8)// bateu no meio
    {
        dir_y = 0;
    }
    if (circle_y <= p1_cima - 16 && circle_y >= p1_baixo)// bateu em baixo
    {
        dir_y = -1;
    }
}
Colisão com o cenário
circle_x = circle_x + dir_x * VEL_X;
circle_y = circle_y + dir_y * VEL_Y; // Calcula a posição da bola

if ((circle_x == 8 || circle_x == 7) &&
(circle_y <= p1_cima && circle_y >= p1_baixo)) // Colisão no jogador 1
{
    chute_jogador_1();
}
if ((circle_x == 115 || circle_x == 114) &&
(circle_y <= p2_cima && circle_y >= p2_baixo)) // Colisão no jogador 2
{
    chute_jogador_2();
}

if (circle_y <= 8) // colisão em cima
{
    dir_y = 1;
}
```

```

if (circle_y >= 91) // colisão em baixo
{
dir_y = -1;
}

```

2.6 Pontuação e Transição de Estados

O jogo também precisa registrar quando um jogador marca um ponto. Isso acontece quando a bola ultrapassa completamente uma das extremidades horizontais da tela. Nesse momento, o placar correspondente é incrementado e a bola retorna ao centro da tela, normalmente com uma direção inicial invertida em relação ao lado do jogador que sofreu o ponto.

Em vez de simplesmente reiniciar a ação imediatamente, é comum colocar uma curta pausa visual, para que o jogador entenda o que aconteceu e se prepare para voltar a jogar. Dica: esse intervalo não deve ser implementado com interrupções completas na execução do programa, mas sim com mecanismos de temporização baseados no tempo acumulado no loop.

Quando algum dos jogadores atinge a pontuação máxima definida, o sistema deve perceber e mudar para a tela de fim de partida, onde se apresenta o vencedor e se aguarda uma nova instrução do usuário para recomeçar ou voltar ao menu

```

// Bateu no gol atrás do jogador 2
if (circle_x >= 120)
{
    dir_x = -1;
    dir_y = 0;
// Aumenta pontuação do jogador 1
    p1_pontuacao++;
// Verifica vencedor
    if (p1_pontuacao == 5)
    {
        estado = VITORIA;
        jogador_vitorioso = 1;
    }
// Posiciona bola no centro do mapa

```

```

        circle_x = TV.hres() / 2;
        circle_y = TV.vres() / 2;
    }
// Bateu no gol atrás do jogador 1
if (circle_x <= 2)
{
    dir_x = 1;
    dir_y = 0;
// Aumenta pontuação do jogador 2
p2Pontuacao++;
// Verifica vencedor
if (p2Pontuacao == 5)
{
    estado = VITORIA;
    jogadorVitorioso = 2;
}
// Posiciona bola no centro do mapa
circle_x = TV.hres() / 2;
circle_y = TV.vres() / 2;
}
}

```

3 Avaliação

Com o desenvolvimento concluído, o aluno deverá submeter o código para que o professor possa avaliar tanto o entendimento e interpretação das fases quanto o código em si. Durante a próxima aula, após avaliar todos os códigos, o professor demonstrará um código exemplo modelo para que os alunos entendam o que poderia ser alterado em casos de erro na interpretação do problema.