POOL PONG

Desenvolvimento de um jogo estilo PONG

1. Referência

O Pong é um jogo inspirado no esporte tênis. Dois jogadores, ou um jogador e a IA, disputam em uma partida pela maior pontuação, cada um controlando uma raquete o objetivo é sempre rebater a bola. A pontuação é realizada quando um dos dois não consegue acertar a bola, neste caso o oponente daquele que errou é quem recebe os pontos.

2. Design

Podemos então utilizar o Pong original como referência e imaginar uma versão única (novas mecânicas, regras e gráficos). Realizar alterações no design original não tem problema desde que a base ainda seja a mesma.

Nesse projeto, então, ao invés de um jogo de tênis podemos imaginar uma partida de vôlei na piscina (de dois jogadores). Há um máximo de três pontos para cada jogador (contaremos como vidas) e, além disso, é necessário que o jogador pressione um botão para rebater a bola.

Mecânicas novas:

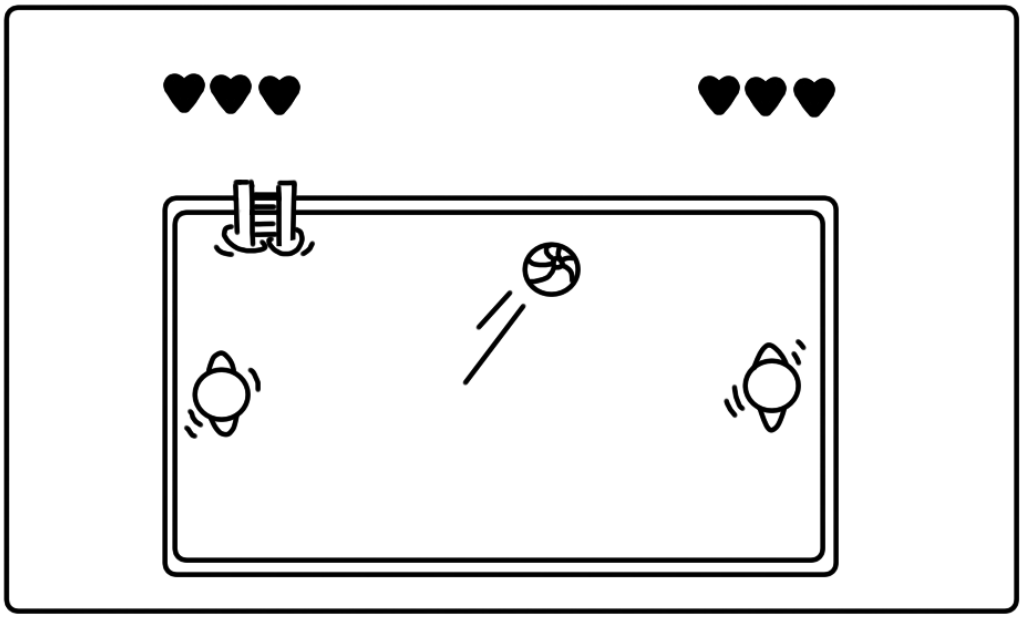
- Sistema de vida

- Sistema de rebater a bola (pressione o botão no momento certo)

- Mapa menor (área jogável não ocupa a tela inteira como o Pong original)

<table>
<tr><td colspan="3">Hud</td></tr>
<tr><td>Player 1</td><td>Piscina

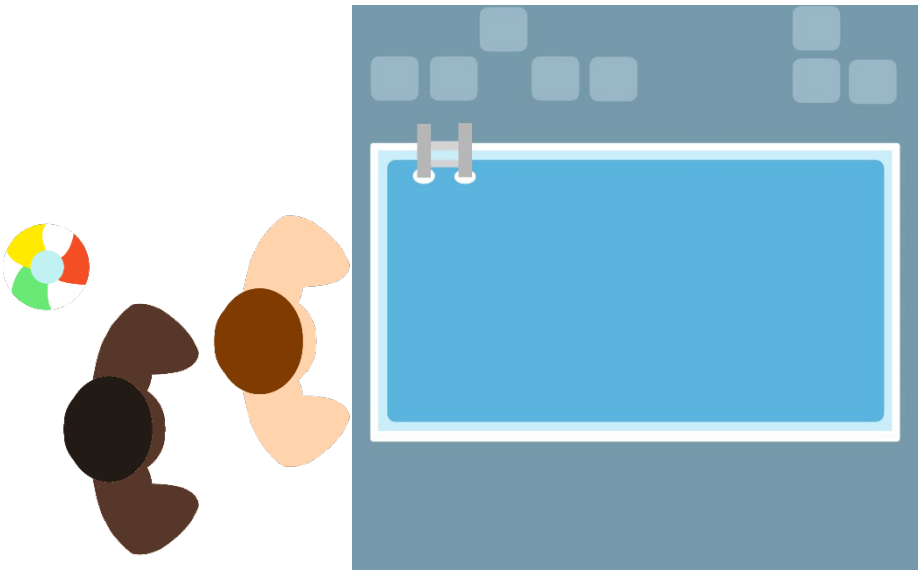
(Área jogável)</td><td>Player 2</td></tr>
</table>



- Imagem conceitual

2.1 Arte

Toda a arte do jogo possuí um design simples (sem sombras, colaração básica), em um estilo top down (visão de cima).



3. Desenvolvimento

O jogo foi criado com a plataforma Scratch, com isso o desenvolvimento foi bem rápido e sem muitas complicações. Contudo vale a pena discutir sobre algumas das etapas do desenvolvimento e as práticas usadas.

3.1 Movimentação da bola

A movimentação da bola é feita utilizando um vetor resultante das velocidades aplicadas no eixo x e y. Com isso obtemos uma movimentação de 8 direções possíveis.

3.2 Limites do Mapa (Colisão)

Diferente do Pong, esse jogo não utiliza toda a tela como arena. Com isso a bola teve que ser configurada para colidir com a beirada de uma área criada por mim.

O código é simples, definimos uma altura y máxima (yMax) e y mínima (yMin), assim se a posição y da bola for maior ou menor que esse limites, sua velocidade no eixo y é invertida. A mesma lógica se aplica ao eixo x.

Y máximo

```
if (bola.y >= yMax || bola.y <= yMin)
        bola.velY *= -1
```

Y mínimo

3.3 IA adversária

Na primeira versão do jogo, havia um segundo jogador, o Pool Pong era obrigatoriamente um jogo de dois jogadores. Contudo, isso limita muito o alcance do jogo (caso a pessoa não tenha alguém para jogar) e por causa disso optei por desenvolver uma IA para enfrentar o usuário.

A parte mais difícil do processo foi o balanceamento, foram necessários muitos ajustes para a IA não ser invencível ou muito fraca, já que o algoritmo usado copia a posição y da bola e fornece ao computador. Por causa disso tive que limitar a IA para seguir a bola somente quando a mesma estivesse muito próxima. (Nem de longe a melhor solução, porém com os valores que eu defini a IA pode ser derrotada).

4. Dificuldades

4.1 Interface de Usuário

Houve alguns problemas e imprevistos para conseguir exibir a vida de cada jogador. Em Pool Pong a vida de cada jogador é exibida com corações. Cada coração equivale a uma vida. Assim a ideia inicial era criar um loop que checasse a quantidade de vida de cada jogador e criasse um coração para cada um de vida.

No entanto, o Scratch utiliza um sistema de clonagem (um pouco confuso), e a lógica que se aplica a um clone não é a mesma do objeto original, o que significa que eu teria que manter dois algoritmos diferentes, o que é bastante inviável. Por isso a solução que escolhi foi a mais básica, porém não a melhor, de simplesmente criar um objeto para cada coração na tela e manter a lógica de cada um individual.

Objeto Coração:

```
var vidaRef, playerRef;
if(playerRef.vida == vidaRef)
        Hide();
// “playerRef” referência do jogador a qual esse coração pertence
// “vidaRef” índice, do coração (se ele é o primeiro, segundo ou terceiro)
```

4.2 Organização do projeto

Conforme o projeto aumenta de tamanho e alterações nos planos são feitas, o código começa ficar cada vez mais complexo. Apesar de ter definido o design, o estilo de arte e o gameplay antes de começar o desenvolvimento, várias mecânicas que pareciam ser interessantes no papel acabam sendo chatas na prática. Assim, todo o código que era estruturado em torno dessas mecânicas é jogado fora e um novo é feito por cima. Isso cria um novo código pouco legível e difícil de modificar.

4.3 Precisão dos hits

A mecânica de rebater a bola parece fácil de fazer, apenas duas checagens (player está colidindo com a bola e pressionando a tecla). O problema é que o código é estruturado com dois if’s agrupados e, por causa disso, se a colisão e o pressionar das teclas não ocorrerem perfeitamente sincronizados o código não funciona:

If(player está colidindo com a bola)

    If(tecla espaço está pressionada)

        Rebater();

//Só funciona se a colisão e o pressionar da tecla ocorrerem no mesmo frame.

5. Feedback

Algumas críticas e avaliações de pessoas que testaram o jogo.

Testadores:

Miguel Brant (irmão): “IA muito previsível, jogo muito curto, a pontuação deveria ser maior. (apenas 3 vidas).”

Déborah Brant (mãe): “muito difícil.” Obs: problema 4.3

Miguel Guerra (pai): “legal.” (não muito específico)

Bugs: Se apertar e segurar espaço muito tarde (com a bola dentro do personagem) ela vai rebater várias vezes e inclusive pode ricochetear para trás.

6. Resultado

Teste e dê sua opinião sobre o jogo!

Link: https://scratch.mit.edu/projects/666662087/