Explicando a física de PONG

Inicialmente, temos o código de movimentação do **Player**

Utilizamos um static body 2d porque queremos que participe do sistema de colisão, mas que não reaja de acordo com a física, como um Rigid Body 2D faria

A direção é definida a partir do input e é normalizada para que não afete a velocidade

Precisamos usar a função de Clamp no player para limitar a sua movimentação vertical. É mais eficiente fazer dessa forma do que fazer uma parede somente para bloquear sua movimentação

```python
position = position.clamp(Vector2.ZERO + Vector2(0, menu_up_height) + object_size/2, screen_size - object_size/2)
```

Temos também a **máquina**, a qual varia de velocidade inicial de acordo com o nível de dificuldade setado pelo jogador

```python
if GameParams.level == 1:
	speed = easySpeed
elif GameParams.level == 2:
	speed = normalSpeed
elif GameParams.level == 3:
	speed = hardSpeed
```

A movimentação da máquina é feita da seguinte forma:

Se a altura da bola na tela é maior do que a minha altura atual (bola acima do sprite da máquina), devo ter minha direção para cima. Caso oposto, devo ter minha direção para baixo.

Se a distância entre a bola e a máquina for maior do que quanto consigo mover no próximo frame (velocidade multiplicada por delta), movo normalmente. Caso oposto, devo mover apenas a distância necessária.

Novamente precisamos limitar a posição para não ultrapassar os limites da tela

```python
if ballY > machineY:
	direction -= 1
elif ballY < machineY:
	direction += 1
if abs(dist) > speed * delta:
```

```
        moveBy = direction * speed * delta
else:
        moveBy = dist

position.y -= moveBy
```

Mas o mais importante na verdade é a bola. Sua direção inicial é aleatória, porém somente entre 4 valores pré-determinados, todos divisíveis por 45 graus. Também normalizo para evitar que seu tamanho seja maior do que o esperado.

```
Python
direction = possibleDirections[randi() % possibleDirections.size()]
direction.normalized()
```

A bola também possui um sprite auxiliar para ressaltar sua direção. A rotação é determinada a partir da inclinação do vetor de direção com o eixo horizontal

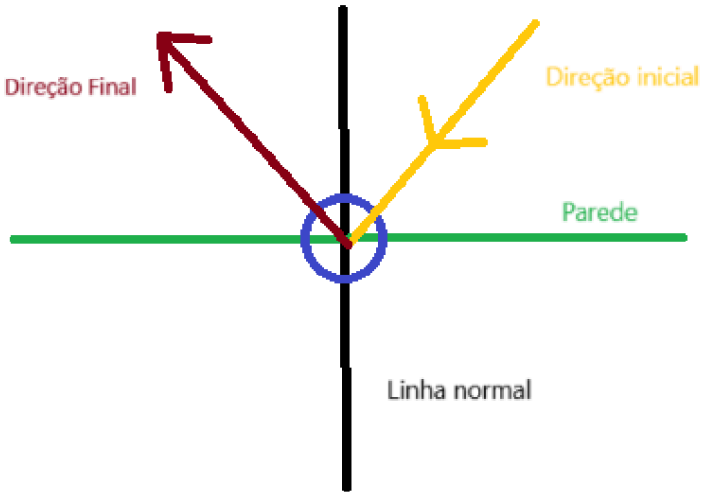
```
Python
pivotSprite.rotation = Vector2(1,0).angle_to(direction)
```

Seria algo semelhante ao desenho abaixo abaixo:

Para a física da bola, usamos um CharacterBody2D, visto que queremos controlar tanto a movimentação quanto a colisão manualmente.

Quando a bola colide na parede, precisamos simplesmente refletir sua direção. Para refletir, precisamos saber do ponto central. Para pegar este ponto, é necessário achar o vetor normal do ponto da colisão e para refletir basta chamar a função de bounce

```
Python
direction = direction.bounce(collision.get_normal())
```



Porém quando a colisão é com o player ou com a máquina, o cálculo deve ser feito de forma mais imprevisível. Quanto mais distante do centro do player a bola estiver, mais vai ser a intensidade da reflexão. Além disso, para não ser extrema, multiplicamos por um valor decimal abaixo de 1 para reduzir seu tamanho, conforme a equação abaixo

```python
newDir.y = (dist / (collider.object_size.y / 2)) * MAX_Y_VECTOR
```