12

Colisão do inimigo com as portas

Agora o inimigo já é bloqueado pelas paredes mas ele continua ignorando as colisões com as portas. Para considerar mais essa colisão, vamos precisar fazer algumas alterações na função **verificaColisaoComObjetos()**. Tínhamos colocado uma verificação separada para o inimigo indicando que ele nunca tinha colisões com objetos. Mais pro começo do curso, gastamos um tempo tornando o código da colisão bem genérico pra gente não ter que ter um monte de `if` verificando se o jogador colidia com chave, porta ou inimigo. Agora vamos precisar mudar um pouquinho o código para considerar que agora pode ser o inimigo ou o jogador colidindo com os objetos e em cada um dos casos queremos ter comportamentos diferentes. Vamos ver o que muda:

- Crie uma constante chamada **JOGADOR** que tenha como valor o texto "JOGADOR" (tudo em maiúsculas). Verifique se a sua constante **INIMIGO** tem como valor o texto "INIMIGO" (tudo maiúsculo também).
- Na função **inicializa()**, crie um atributo chamado **tipo** no **jogador** com o valor `Constantes.JOGADOR`.
- Atualize as funções **criaChave()**, **criaPorta()** e **criaInimigo()** para que todos esses objetos possuam um atributo chamado **colisoes** que tenha dentro dele uma função de colisão para o inimigo e outra para o jogador. Por exemplo, na função **criaPorta()**, queremos ter `local porta = {colisoes = {INIMIGO = fazColisaoDoInimigoComAPorta, JOGADOR = fazColisaoDoJogadorComAPorta}, ...}`. Para alguma colisões, queremos simplesmente deixar o inimigo passar então crie uma função **deixaPassar()** que recebe um **indice** e devolve sempre **false**. Use essa função para quando o inimigo colidir com uma chave ou com um outro inimigo.
- Atualize a função **verificaColisaoComObjetos()** para que quando ocorra uma colisão ela escolha a função de colisão do objeto a usar com base no tipo da personagem. Você deve ter algo como: `local funcaoDeColisao = objeto.colisoes[personagem.tipo]`.

Com isso, o inimigo já deve ser bloqueado pelas portas do calabouço. Ainda resta um problema que identificamos há algumas aulas atrás. Quando o inimigo encosta no jogador, se o jogador estiver parado não temos colisão! Isso acontece porque só verificamos a colisão entre os objetos se o jogador estiver se movendo. Para resolver só precisamos checar também a colisão na posição atual do jogador.

- Na função **atualiza()**, chame a função **verificaColisaoComObjetos()** passando o **jogador** como primeiro parâmetro e um objeto com **x** e **y** iguais às coordenadas atuais do jogador. Isso vai fazer a verificação de colisão na posição atual do jogador.

Legal, agora só precisamos deixar o inimigo um pouco mais devagar pra que o jogador tenha alguma chance de fugir:

- Modifique a função **atualizaInimigo()** para que o inimigo se mova apenas 0.5 pixel por vez ao invés de 1.

Pronto! Com isso concluímos a primeira versão do nosso inimigo! Teste bastante seu código pra ter certeza que o inimigo está seguindo todas as regras do jeito que a gente definiu.