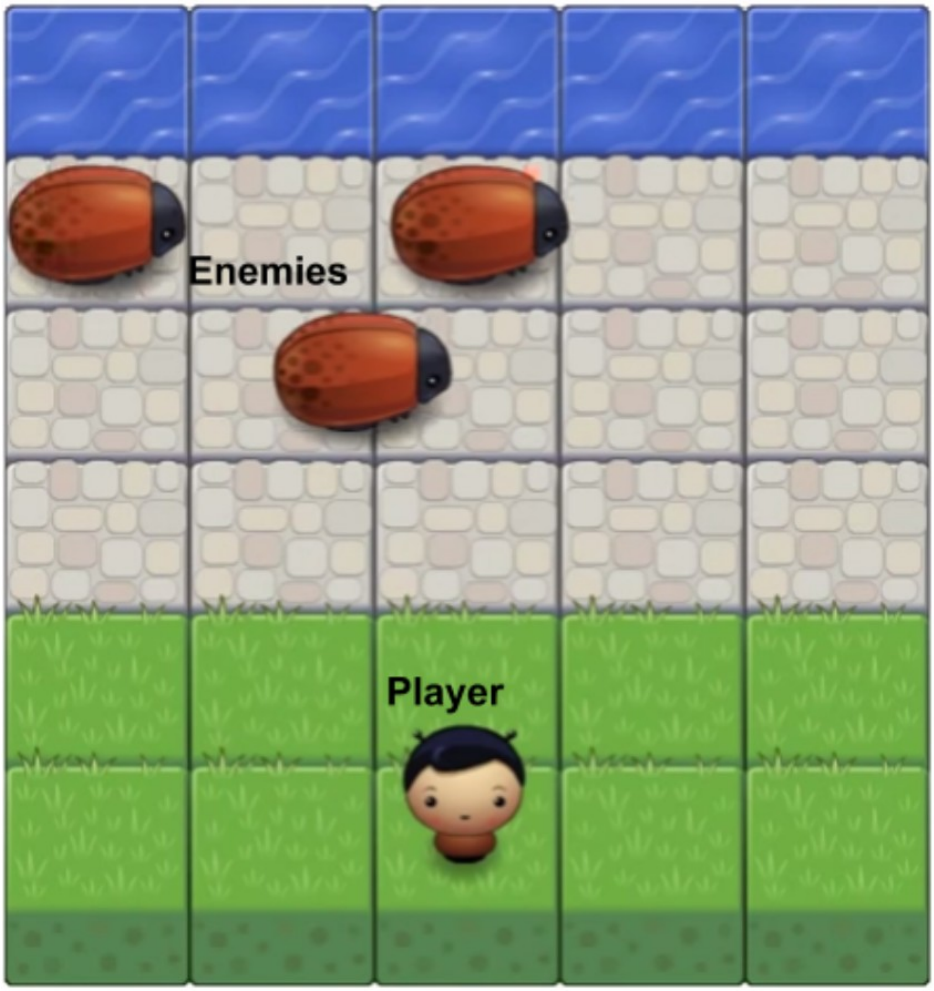
Neste trabalho você construirá uma versão simples do jogo: http://froggerclassic.appspot.com/ .

Descrição do jogo -

Funcionalidade básica -
Neste jogo você tem o jogador e os inimigos(insetos). O objetivo do jogador é alcançar a água, sem colidir com nenhum dos inimigos. O jogador pode se mover para a esquerda, para a direita, para cima e para baixo. Os inimigos se movem em velocidades variáveis na parte do bloco pavimentado da cena. Uma vez que o jogador colide com um inimigo, o jogo é reiniciado e o jogador volta para o quadrado inicial. Quando o jogador atinge a água, o jogo é ganho.



Funcionalidade adicionais -

Além da funcionalidade básica, você pode adicionar mais funcionalidades interessantes ao seu jogo. Por exemplo, aqui estão alguns recursos adicionais que você pode adicionar:

- Seleção do jogador: permite ao usuário selecionar a imagem do personagem antes de iniciar o jogo. Você pode usar as diferentes imagens de personagem fornecidas na pasta de imagens (vamos ver isso abaixo).
- Pontuação: você pode implementar uma pontuação para o jogo. Por exemplo, a pontuação pode aumentar a cada vez que o jogador atinge a água, e pode ser redefinida para 0 quando ocorrer uma colisão (ou pode ser reduzida).
- Colecionáveis: você pode adicionar pedras preciosas ao jogo, permitindo ao jogador colecioná-las para tornar o jogo mais interessante.
- Ou qualquer outra coisa que achar interessante.

Você não precisará criar o jogo do zero. São disponibilizados os ativos de arte e a engine de jogo para você.

O repositório contém pastas css, imagens e js, além de um arquivo index.html. Depois de ter baixado os arquivos, você terá que editar app.js para construir o jogo.

A pasta css contém um arquivo style.css que você não precisa editar. A pasta images contém os arquivos de imagem png, que são usados ao exibir o jogo. As imagens para o jogador e o personagem inimigo serão carregadas desta pasta. A pasta js também contém a engine necessária para executar o jogo e um arquivo resources.js. Você não precisa editar esses arquivos. index.html é o arquivo que carregará o jogo.

Dentro do arquivo app.js, você precisará implementar as classes Player e Enemy. Parte do código para o Inimigo é fornecida a você, e você precisará completar o seguinte:

- A função Enemy, que constrói um inimigo:
  - Carregando a imagem ao definir this.sprite com a imagem apropriada da pasta de imagens (já fornecida).
  - Definindo a localização inicial do inimigo (você precisa implementar).
  - Definir a velocidade do inimigo (você precisa implementar).
- O método de atualização para Enemy:
  - atualiza a localização do inimigo (você precisa implementar).
  - Manipulador de colisão com o jogador (você precisa implementar).
- Você pode adicionar métodos a Enemy se achar necessário.

Você também precisa implementar a classe Player, e pode usar a classe Enemy como exemplo de início. Você deve implementar:

- A função Player, que constrói o jogador:
  - Carregando a imagem ao definir this.sprite com a imagem apropriada da pasta de imagens (já fornecida).
  - Definir a localização inicial do jogador.
- Método de atualização do jogador.

- Método de renderização do jogador (pode usar do Enemy).
- Método handleInput, que deve receber a entrada do usuário, allowedKeys (a tecla pressionada) e mover o jogador de acordo com a entrada. Em particular:
  - Tecla esquerda deve mover o jogador para a esquerda, direita para a direita, para cima deve mover o jogador para cima e para baixo deve mover o jogador para baixo.
  - Lembre-se de que o jogador não pode sair da tela (assim, você precisará verificar isso e manipular adequadamente).
  - Se o jogador atingir a água, o jogo deve ser reiniciado, movendo o jogador de volta para o local inicial (você pode escrever um método separado de reset para lidar com isso).
- Você pode adicionar métodos a Player se achar necessário.

Depois de concluir a implementação do Player e do Inimigo, você deve instanciá-los:

- Criando um novo objeto Player.
- Criando vários novos objetos Enemy e colocando-os em uma matriz chamada allEnemies.

Se você quiser, pode adicionar funcionalidade adicional ao jogo. Você pode adicionar mais código ao arquivo app.js e às classes Enemy e Player para conseguir isso.