

Exercício 3 – Filas de Comandos

1. Classe FollowerObject: Seguindo comandos do Mouse.

FollowerObject : GameObject
<ul style="list-style-type: none">- sprite : Sprite*- speedX, speedY : float- coordinatesQueue : std::queue<Point>
<ul style="list-style-type: none">+ FollowerObject(sprite : Sprite*, x : float, y : float)+ update(dt : int) : void+ render(camera : float, cameraY : float) : void+ renderQueueLines(camera : float, cameraY : float) : void+ enqueueCommand(pos : Point<float>) : void

A classe FollowerObject será usada para criar um objeto que se move a partir de comandos do mouse.

Sobre os atributos da classe:

- sprite: Sprite*

Armazena a sprite usada na renderização do objeto.

- speedX, speedY : int

Guardam as posições do mouse.

- coordinatesQueue : std::queue<Point>

Será usada para guardar todos os comandos dados ao objeto.

Alternativamente, pode-se usar 2 listas de float, uma para X e outra para Y.

Sobre os métodos da classe:

- + FollowerObject(sprite : Sprite*, x : float, y : float)

Construtor da classe, inicializa a sprite do objeto e sua posição inicial.

+ update(dt : int) : void

O método update deve:

- Checar se a lista de coordenadas não está vazia (ou seja, se tem algum comando de movimento para executar).
- Caso tenha, o objeto deve calcular sua velocidade para se movimentar até o objetivo. Seu deslocamento deve ser dado de acordo com dt, como explicado no trabalho 2. Idealmente este movimento deve ser suave e proporcional em cada eixo (de modo que a nave se movimente efetivamente em linha reta).
- **Dica:** Utilize a distância até o objetivo para calcular a velocidade proporcional em cada eixo. Utilize a posição dos pontos para determinar o sentido da velocidade.
- Testar se a nave já chegou no objetivo (**aproximar!**). Caso tenha, deve ficar parada lá enquanto não houver novos comandos.

+ render(camera : float, cameraY : float) : void

Renderiza o objeto pelo centro. Ou seja, caso o objeto esteja em (0,0) seu centro deve estar nessa posição, e não o canto superior esquerdo da imagem.

+ renderQueueLines(camera : float, cameraY : float) : void

Este método deve mostrar na tela linhas que correspondem aos comandos enfileirados (caso haja algum). Para isso, deve ser usado o método `SDLBase::drawLine`, fornecido nos arquivos do laboratório.

Dica: Não é possível iterar na fila! Para acessar todos os elementos da fila será necessário removê-los e colocá-los em uma fila auxiliar, um a um. Uma boa estratégia é remover os elementos, renderizá-los, e inseri-los em uma lista auxiliar e, por fim, atribuir essa lista de volta para a original.

+ enqueueCommand(pos : Point<float>) : void

Coloca a posição pos na fila. Alternativamente, pode-se usar 2 listas, uma para X e outra para Y, e passar ambos os parâmetros como argumento.

Alterações na Game Engine:

SDLBase:

- Acrescentar o método `drawLine`;
- Alterar o nome do executável.

GameManager:

- Instanciar um `followerObject`.
- Acrescentar o `update`, `render`, `renderQueueLines` e `delete` do novo objeto em seus devidos lugares.
- Na `processEvents`, testar o clique do botão direito do mouse (botão 3) e enfileirar esse novo comando no `followerObject`.
- Os planetas devem continuar sendo destruídos com o clique do botão esquerdo.