Uma imagem com Tipo de letra, Gráficos, design gráfico, captura de ecrã

Descrição gerada automaticamente

**Relatório Sprint 3**

**Turma 3DC-Grupo 14**

1210825 - Pedro Ferreira

1211134 - Pedro Viana

1211436 - David Pinho

1211439 - Pedro Costa

**Professor:**

Eduardo Tovar, EMT

**Unidade Curricular:**

Sistemas Gráficos

**Breve descrição das tarefas**

Identificação das tecnologias utilizadas

Interação adotada (identificação das teclas utilizadas e das ações do rato, se aplicável)

Representação gráfica em 3D

**601 – O formato base do mapa que descreve a planta de um piso de um edifício do campus (um ficheiro/piso/edifício) deve ser complementado com informação do módulo de gestão de campus: dimensão do piso, disposição das paredes, localização das portas das salas/gabinetes, dos acessos e dos elevadores, etc.**

O formato base do mapa que descreve a planta de um piso de um edifício do campus está representado num ficheiro com extensão json e contém as seguintes informações do módulo de gestão:

* Dimensão do piso – representado pela variável size já existente no ficheiro com extensão json do projeto do labirinto
* Disposição das paredes – mantiveram-se os números já existentes na matriz do projeto do labirinto para representar as suas localizações
* Localização das portas – foram acrescentados os números 4 e 5 à matriz para representar portas norte e oeste, respetivamente
* Localização dos acessos – variável accessToBuilding para representar a localização dos acessos aos corredores de passagem, apenas nos ficheiros dos pisos que têm estes acessos
* Localização dos elevadores – foram acrescentados os números 6 e 7 para representar as portas dos elevadores norte e oeste, respetivamente, e as variáveis elevatorDoorLocation, elevator e elevatorDirection para representaram a localização dos elevadores

OU

Para realizar esta user story foram criados ficheiros com extensão json para todos os pisos de cada edifício, a partir do ficheiro com extensão json do projeto do labirinto. Na matriz manteve-se os números já existentes para representar a localização das paredes pelo que apenas foram acrescentados os números 4 e 5 para representar portas norte e oeste, respetivamente, e 6 e 7 para representar as portas dos elevadores norte e oeste, respetivamente. Além desta alteração, foram adicionadas as variáveis elevatorDoorLocation, elevator e elevatorDirection para representaram a localização dos elevadores. Também foi acrescentada a variável accessToBuilding para representar a localização dos acessos aos corredores de passagem. A dimensão do piso é representada pela variável size já existente no ficheiro com extensão json do projeto do labirinto.

**602 – O formato base do mapa que descreve a planta de um piso de um edifício do campus (um ficheiro/piso/edifício) deve definir, além da grelha matricial, a identificação dos ficheiros contendo as texturas adequadas às paredes dos diversos pisos, bem como os modelos 3D representativos das portas e dos elevadores.**

**603 – Permitir a visualização em 3D do interior dos pisos dos edifícios do campus (um piso de cada vez): chão, paredes, portas e elevadores.**

**604 – Criar uma interface com o utilizador (GUI) que permita selecionar o edifício e o piso que se pretende ver em cada momento.**

Falta explicar pastas criadas e variáveis parameters

A função changeBuildingParameters valida os parâmetros que permitem carregar o novo piso e caso sejam válidos chama a função createMaze. A função createMaze remove o piso atual da cena, define gameRunning como false e cria um novo piso com base nos parâmetros fornecidos, por fim atualiza a referência do piso atual para o novo piso.

**605 – Importar o modelo 3D de um robot e permitir a sua visualização numa dada localização do piso selecionado.**

**606 – Permitir a navegação interativa (com o recurso ao teclado) do robot pelos diversos pisos dos diversos edifícios do campus.**

Navegação e animação

**608 – Ao navegar interactivamente e ao entrar num elevador, deve ser possível escolher o piso a que se pretende aceder e transitar para esse piso.**

Para implementar esta US fizemos um método onde vai dar return de true se houver colisãao com o robot, este está definido através do elevator.possition, que fomos buscar à description que vem através do argumento do maze.

Esta função é recebida pelo ThumbRaiser, passando por parâmetros o valor da posição do robot.

De seguida, dentro do if, ou seja, se houver colisão, vai mostrar a animação, que vai ser já explicada e vai fazer uma sequência de ações, vai pôr o HTML do popup visível e vai pausar o jogo.

Porém é importante destacar o parâmetro playerPosition, uma vez que neste caso, esta variável é utilizada para que o robô inicie na posição do elevador do novo edifício.

**609 – Animar o movimento automático do robot de um dado ponto de partida a um dado ponto de chegada, de acordo com a informação disponibilizada pelo módulo de planeamento de percurso.**

Para esta User Story, o robot move-se automaticamente pelas células do mapa 3D correspondente, o caminho de células é recebido pelo módulo de Planeamento que fornece os valores do x e y de cada célula por ordem. Para implementação temos de ter atenção também que o robot pode movimentar-se para todo o lado o que significa que podemos ter células nas diagonais.

A nossa ideia principal para esta tarefa foi implementar aos pares de células de cada vez, em que o programa verificava a posição da célula seguinte consoante a respetiva célula atual, se esta á direita, em cima, em baixo, a esquerda ou numa das diagonais. Sabendo a posição da célula seguinte para onde o robot tem de se deslocar, sabemos também o seu respetivo ângulo, visto que temos de rodar o robot para que fique de frente para a posição onde se tem de deslocar para que não ande de lado. Podemos pensar nisto como vermos o mapa de cima e colocarmos um círculo dividido em 8, nos seus principais ângulos (0, 45, 90, 135, 180, 225, 270, 360).

Uma imagem com captura de ecrã, texto

Descrição gerada automaticamente

Primeiro rodamos então o robot para o ângulo calculado anteriormente, de seguida fazemos o robot deslocar-se em frente até alcançar o centro da célula seguinte, isto é conseguido através de um método chamado **foundCell (posicaoAtual, posicaoFinal)** onde fornecemos a posição atual do robot e a posição da célula de destino**,** esta função é verificada em cada update, caso haja sucesso começa-se a tratar-se do próximo par de células e repete-se o mesmo processo. No caso de insucesso da função anterior, o robot continua a deslocar-se em frente através da adição de um novo X e Z com o vector3 do Three.js, faz-se a animação de movimento do robot e atribui-se a nova posição.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

**Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente**

**610 – Ao navegar de forma automática com base no planeamento de percurso, a passagem entre edifícios deve ter feedback visual adequado.**

**611 – Ao navegar de forma automática com base no planeamento de percurso, a utilização de elevadores deve ter feedback visual adequado.**

Nesta US, usámos uma animação que estava por automática no projeto para simular o robot a tocar no botão do elevador para o usar, desta maneira usamos um TimeOut para distinguir esta animação do PopUp criado quando há colisão com o elevador.