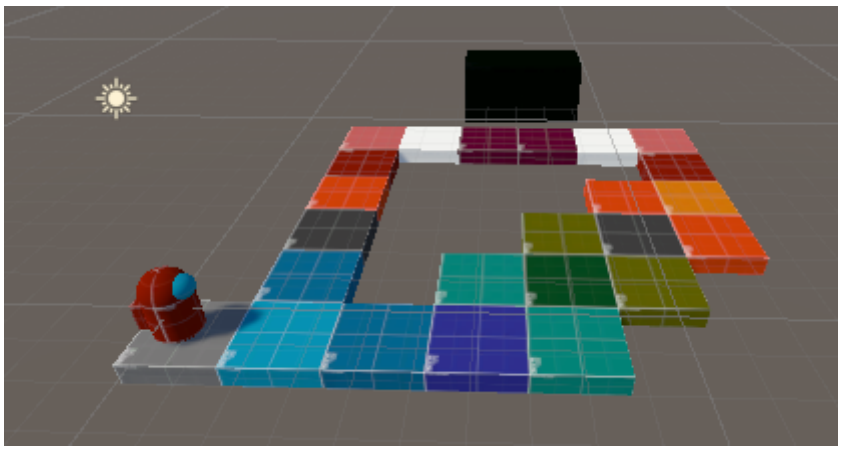
Projeto Q-learning:

O que vai ser dado:

Para esse projeto, será disponibilizado: (1) um arquivo .exe que contém o jogo; (2) um projeto em python no GitHub que contém um arquivo chamado connection.py (i.e., a conexão com o servidor local do jogo) e um outro arquivo chamado client.py, que será onde vocês irão implementar o algoritmo de vocês.

Objetivo do jogo:

Neste jogo controlamos o personagem Amongois, que terá que passar por diversas plataformas para chegar no seu objetivo final que é o bloco preto. Para isso ele pode fazer 3 movimentos, sendo eles, girar para a esquerda, girar para a direita e pular para a frente.



Objetivo do projeto:

O objetivo do projeto é implementar o algoritmo Q-Learning para aprender o trajeto que deve ser tomado pelo Amongois no jogo.

Como conectar o seu algoritmo ao jogo:

Para que haja uma comunicação entre o algoritmo e o jogo, vocês irão inicialmente importar o **connection.py** para o **client.py** (local onde vocês irão implementar o algoritmo). Para vocês iniciarem a conexão, devem chamar a função **connect()** que irá retornar o socket com a conexão. O **connect()** recebe a porta utilizada pelo executável como argumento.

Após realizada a conexão, a comunicação irá se dar pela função **get_state_reward().** Essa função recebe a ação que deve ser feita pelo personagem e o socket recebido na função **connect()**, e terá como retorno, **o estado atual** e a **recompensa**.

Exemplo:


```
client.py M
Qlearning > client.py > ...
You, 16 seconds ago | 1 author (You)
#Aqui vocês irão colocar seu algoritmo de aprendizado
import connection as cn

s = cn.connect(2037)

estado, recompensa = cn.get_state_reward(s, "jump")
```

Como é o formato das informações enviadas para o jogo:

A informação para cada ação é representada por uma string correspondente:

"left" = Girar para a Esquerda

"right" = Girar para a Direita

"jump" = Pular para a Frente

Como jogar

Você pode jogar manualmente ou você pode controlar o jogo da forma descrita acima.

- **Jogando manualmente**

  É só utilizar as setas de direita e esquerda para mudar a direção e a barra de espaço para pular para a frente.

Obs.: tecnicamente você pode usar a seta para a frente para andar nesse sentido, mas ela anula o não-determinismo dessa ação.

- **Atalhos de teclado**
  - 1: aumenta a velocidade do Amongois (cuidado para não acelerar muito e exigir demais da sua máquina).
  - 2: diminui a velocidade do Amongois.
  - 3 a 7: tamanhos progressivamente menores de tela.

Como é o formato das informações recebidas pelo jogo:

- **Para o Estado:**

  Um estado é representado como um vetor binário que concatena a informação de que plataforma o personagem se encontra e para qual sentido ele está virado. Como são 24 plataformas possíveis, serão usados 5 dígitos em binário para essa representação. Para o sentido, por sua vez, serão usados dois dígitos, como segue:

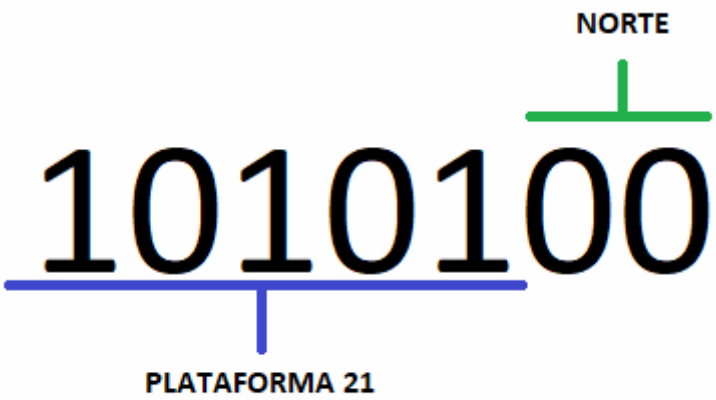
  00 = Norte

  01 = Leste

  10 = Sul

  11 = Oeste

  Usando a função **get_state_reward(),** o servidor envia o estado para o cliente. Abaixo, temos um exemplo de um estado enviado:



- **Para a Recompensa:**

  A recompensa será um número inteiro negativo que irá variar de -1 a -14, considerando o estado resultante do personagem, retornado pela ação **ANTERIOR.**

Entrega do projeto:

Para a entrega do projeto, vocês devem enviar dois arquivos:

1) O arquivo do cliente.py com o algoritmo de vocês.
2) Um arquivo .txt que tenha a Q-table de vocês

Observações:

Tentem deixar o arquivo do cliente.py de forma organizada e bem documentada. Quanto melhor documentado/organizado, mais fácil será para entender o que foi escrito e, com isso, melhor e mais rápida será a correção.

O arquivo com o Q-table deve **CONTER SOMENTE OS DADOS ORDENADOS DE ACORDO COM O ESTADO CORRESPONDENTE**, ou seja, **NÃO É PARA CONTER O TÍTULO DAS COLUNAS E NEM O NÚMERO DAS LINHAS.**

A ordem das colunas na Q-Table deve ser **[Giro para Esquerda, Giro para Direita, Pulo para Frente],** respectivamente.

Por fim, a Q-table deve conter **TODOS OS POSSÍVEIS ESTADOS**. Como são 24 plataformas para 4 direções, temos **96 estados diferentes**.

EXEMPLO: ver arquivo resultado.txt no repositório do github supracitado.