Instituto Superior de Engenharia de Lisboa (ISEL)

Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores (DEETC)

LEIM

Licenciatura em Engenharia Informática e Multimédia

Interação em Ambientes Virtuais

Projeto Final

Miguel Silvestre (45101)

Pedro Dias (45170)

Docente

Professor Arnaldo Abrantes

Julho, 2022

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Este trabalho visa criar um jogo em Unity [Technologies, 2005], utilizando os conhecimentos adquiridos ao longo do semestre, nomeadamente geração procedimental, aprendizagem automática e interação Pessoa-Máquina.

O projeto consiste na criação de um mundo Minecraft [Studios, 2011], onde agentes são treinados a desempenhar determinadas tarefas e o jogador consegue interagir com o ambiente, modificando o curso dos acontecimentos.

Este documento está organizado da seguinte forma:

TODOTODOTODO esperar final e/ou confirmação led TODOTODO-TODOTODO

Capítulo 2

Ambiente

Neste capítulo será ilustrado a forma como o terreno é criado, os tipos de blocos existentes no jogo, o aspeto do céu e os sons que gera.

2.1 Terreno

A construção do terreno deste projeto é semelhante à do primeiro trabalho prático. Para ser possível termos um mundo infinito, é necessário este estar otimizado, de forma a diminuir o peso computacional. Assim, um mundo é criado continuamente em torno do agente à medida que este se movimenta e por conseguinte, parte do mundo é destruído quando este se afasta consideravelmente.

Como dito anteriormente, este projeto visa a criação de um mundo *Minecraft* [Studios, 2011], ou seja um mundo baseado em cubos. Neste ambiente, existem faces que não são visíveis ao jogador, mas que ao serem criadas e renderizadas tornam a execução mais lenta. Assim, de forma a reduzir o número de triângulos, vértices, etc, e melhorarmos o desempenho, não processámos estas faces.

2.1.1 Ruído de Perlin

O ruído de Perlin [Perlin,] é uma técnica muito usada em jogos e utiliza uma série de números parcialmente aleatórios com o objetivo de imitar objetos naturais, como o sol, nuvens, animações, terrenos, entre outros, e permite o controlo de elementos em pequena e grande escala. Utilizamos este ruído

de Perlin na nivelação do nosso terreno, ou seja quanto maior for a sua regularização, mais nivelado ficará o terreno.

TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO

METER VALOR QUE USAMOS E PORQUE - ainda não está bem definido so yea

TODO TODO TODO TODO TODO TODO TODO TODO TODO TODO

2.1.2 Algoritmo Flood Fill

De modo a ser possível construir os diferentes tipos de blocos, utilizamos o algoritmo de Flood Fill [Flo,] que os desenha recursivamente. Em termos gerais, este algoritmo determina a área conectada num nó, dado em um vetor multi-dimensional de nós. Isto possibilita-nos a gerar mais do que um bloco do mesmo tipo, criando assim blocos vizinhos.

2.2 Blocos

tipos de blocos (grass, dirt, stone etc)

mostrar também texturas e materiais

2.3 Skybox

mostrar aspeto final da skybox

2.3.1 Mundo Escondido

De modo a dar uma funcionalidade extra ao jogo, existe um mundo que, inicialmente, se encontra escondido ao jogador. Para ser possível interagir com este mundo, é necessário colecionar 20 blocos de terra, 10 de pedra e 6 de diamante. Neste mundo é gerado de uma forma mais aleatória, onde octaves, smooth e persistence tomam diferentes valores e altera a skybox.

2.4 Som

TODO TODO TODO Falar do som, música ambiente + efeitos sonoros se metermos TODO TODO TODO

2.5 B-spline

De forma a preencher e embelezar o ambiente, foram colocados dragões a voar pelo céu. Isto foi possível com a inclusão de B-splines que definem a trajetória do dragão. Assim, enquanto o jogo estiver a decorrer, eles seguirão o trajeto e dão a sensação de vida ao jogo.

TODO TODO meter imagens TODO TODO

Capítulo 3

Agentes

Aqui são identificados os agentes autónomos do projeto e a forma como foram treinados.

3.1 Aves

Existem dois agentes autónomos, nomeadamente a presa e o predador. Ambos correspondem a aves, um pintassilgo e uma águia. O objetivo do predador é apanhar a presa, e da mesma forma, o pintassilgo tenta fugir da águia.

O seu propósito é dar uma maior diversidade ao ambiente, permitindo ao jogador focar-se mais no mundo Minecraft.

TODOTODOTODOTODOTODO Meter imagens TODOTODOTODO-TODOTODO

3.2 ML-Agents

O Unity Machine Learning Agents Toolkit (ML-Agents) [Unity,] consiste num projeto de código aberto, onde jogos e simulações podem servir como ambientes para efetuar o treino de agentes inteligentes.

Os agentes autónomos foram assim treinados com esta ferramenta. Recorrendo a uma aprendizagem por reforço, os agentes conseguem identificar (através de Raycasts) o seu predador (no caso do pintassilgo) ou a sua presa (se for uma águia).

A metodologia de treinamento foi bastante simples: quando as diferentes aves se tocam, a águia recebe uma recompensa positiva e o pintassilgo uma

negativa. o predador continuamente recebe uma penalização mínima, ao contrária da presa que é continuamente beneficiado.

TODO TODO TODO meter aqui o do TP2 TODO TODO TODO

3.3 Ficheiro de Configuração

De modo a treinar um agente, é necessário um ficheiro ". yaml" de configuração, onde é possível definir inúmeros parâmetros diferentes. O ficheiro utilizado encontra-se na figura 3.1.

Devido ao número total de parâmetros possíveis definir ser muito elevado, iremos dar uma breve explicação dos que foram utilizados neste trabalho:

- batch size - número de experiências de cada iteração.
- buffer size - número de experiências colecionadas antes de atualizar o modelo;
- learning rate - corresponde à taxa de aprendizagem;
- Beta - corresponde à força de regularização entrópico. Quanto maior for este valor, mais ações aleatórias são tomadas;
- Epsilon - influencia a rapidez com que a política evolui durante o treino. Estamos a utilizar o valor por omissão 0.2;
- Lambd - parâmetro de regularização com valor por defeito de 0.95. Significa o quão um agente depende dos valores atuais. Se este parâmetro possuir um valor baixo, significa que dependerá mais, enquanto que valores altos fazem com que dependa mais nas recompensas recebidas no ambiente
- Num epoch - também o valor por defeito 3, corresponde ao número de passagens a serem feitas pelo buffer ao realizar a otimização de descida de gradiente
- Learning rate schedule - para o nosso tipo de treino (ppo) o valor de defeito é linear. Este decrementa o learning rate de forma linear, chegando a 0 no "max steps";

- Normalize - valor booleano que diz se a normalização é aplicada ao vetor de observações de entrada;
- Hidden units - o range varia tipicamente entre 32 e 512, no nosso caso estamos a usar o valor 128 que é o valor por omissão. Este, corresponde ao número de unidades que estão completamente conectadas à camada da rede neural;
- Num layers - número de camadas escondidas na rede neural (valor por defeito é 2);
- Gamma - fator de desconto de recompensas futuras a vir do ambiente;
- Strength - fator a ser multiplicado pela recompensa dada pelo ambiente;
- keep checkpoints - número máximo de "pontos de controle" de modelos a manter. Estamos a usar o valor por defeito (5);
- Max steps - número total de passos que devem ser tomados no ambiente antes do treino terminar;
- Time horizon - Estamos a usar o valor por defeito (64). Significa o número de passos de experiência que cada agente coleciona antes de adicionar ao buffer;
- Summary freq - número de experiências que necessitam ser coletadas antes de gerar e mostrar as estatísticas de treino;

```
behaviors:
  Presa:
    trainer_type: ppo
    hyperparameters:
      batch_size: 1024
      buffer_size: 10240
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 256
      num_layers: 1
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 5
    max_steps: 2000000
    time_horizon: 64
    summary_freq: 10000
  Predador:
    trainer_type: ppo
    hyperparameters:
      batch_size: 1024
      buffer_size: 10240
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: false
      hidden_units: 256
      num_layers: 1
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
    keep_checkpoints: 5
    max_steps: 2000000
    time_horizon: 64
    summary_freq: 10000
```

Figura 3.1: Ficheiro de Configuração Presa e Predador

Capítulo 4

Interface

Este capítulo ilustra a interface do jogo.

4.1 Menu

TODO TODO TODO mostrar o menu inicial e explicar a cena da voz TODO TODO TODO

4.2 Menu de Jogo

mostrar o menu do jogo, os blocos e as opções etc

Capítulo 5

Interação

Este capítulo trata a forma como o jogador interage com o ambiente, nomeadamente recorrendo ao teclado e microfone.

5.1 Rato e Teclado

Com o movimento do rato, o jogador consegue ajustar a câmara e, caso um dos botões do rato seja premido (direito ou esquerdo), um cubo será criado ou destruído, respetivamente.

Caso o jogador possua quatro blocos de pedra, ele tem a possibilidade de criar uma picareta premindo a tecla M. Para facilitar a movimentação entre o mundo, consegue saltar clicando no espaço do teclado.

5.2 Microfone

Para ser possível interagir com o jogo através de áudio, utilizou-se um exemplo dado na aula, nomeadamente se o microfone detetar um som acima de 40 decibéis e a concentração no pico da frequência for superior a um dado valor então assume-se que é um assobio e o jogador irá andar para a frente.

Assobiar irá trocar o tipo de bloco que o jogador consegue construir. Estes alteram-se num ciclo predeterminado entre TODO TODO TODO terra pedra e ?? TODO TODO TODO

Capítulo 6

Conclusão

Pode-se concluir que os objetivos do projeto foram concluídos. Todos os temas lecionados ao longo do semestre foram agregados, nomeadamente geração procedimental com a construção de um mundo *Minecraft* [Studios, 2011], aprendizagem automática onde diferentes agentes foram treinados e interação Pessoa-Máquina de modo a ser possível ao utilizador interagir e alterar o curso dos acontecimentos.

Assim, foi possível consolidar os conhecimentos adquiridos ao longo das aulas, que já tinham sido postos em prática nos mini-projetos, em algo mais complexo e onde, havendo inúmeras funcionalidades adicionais que possam ser criadas, seja possível explorar estes conceitos livremente.

TODO TODO TODO TODO TODO TODO TODO TODO TODO

não sei bem o que escrever mais, o segundo parágrafo já foi bue forçado xd talvez falar de dificuldades, cenas assim

TODO TODO TODO

Bibliografia

[Flo,] Flood fill. https://en.wikipedia.org/wiki/Flood_fill.

[Perlin,] Perlin, K. Perlin noise. https://en.wikipedia.org/wiki/Perlin_noise.

[Studios, 2011] Studios, M. (2011). Minecraft. https://www.minecraft.net.

[Technologies, 2005] Technologies, U. (2005). Unity. https://unity.com/.

[Unity,] Unity. Ml-agents. https://github.com/Unity-Technologies/ml-agents.