Protokol k semestrální práci z BI-ZUM

FIT ČVUT. LS 2019/2020

Jméno studenta: Petr Šejvl

Username: sejvlpet

Název semestrální práce: Multi-pacman

OSNOVA

1. Zadání semestrální práce.

Vyvinout řídící algoritmus pro hru Pacman, ve které hraje více pacmanů najednou.

2. Stručný rozbor, analýza problému/zadání.

Pro hru pacman je třeba vyvinout algoritmus, který bude ovládat více agentů. Agenti se snaží co nejrychleji sníst všechny kuličky a hru tak vyhrát, zároveň se ale musí vyhýbat duchům, kteří naopak mohou sníst je. Duchové se pohybují v okolí pacmana (okolí je pro duchy voleno náhodně v závislosti na velikosti hřiště) hladově, jinak náhodně.

3. Výběr metody.

Vzhledem k větvení herního stromu (v každé situaci je až 4 ^ počet pacmanů možných tahů) byla zvolena Monte Carlo Tree Search metoda.

4. Popis aplikace metody na daný problém.

Na začátku se vytvoří instance třídy MCTS, která zajišťuje výběr pohybu pacmanů. Ta si do členské proměnné root vytvoří instanci třídy Node, která představuje jeden uzel z herního stromu. Metodou play třídy MCTS poté vybere a zahraje nejvhodnější tah. Hledání nejvhodnějšího tahu je omezeno časově.

Výběr nejvhodnějšího tahu probíhá vždy tak, že se zavolá findMove metoda na root Nodu. Tato metoda najde nejlepšího kandidáta pro další testování. Tuto selekci jsem nakonec zvolil vlastní a to takovou, která zaručí že se vždy přednostně prozkoumají zatím nenavštívené vrcholy. Pokud byly všechny navštíveny, vybere se další kandidát náhodně s výrazným zohledněním jeho předchozích výsledků.

Pokud byl vybraný kandidát již inicializovaný, zavolá se na něm opět metoda findMove a prozkoumáme se tak další z jeho dětí. Výsledek takové hry se pak uloží jak na aktuálním kandidátovi, tak vybraném dítěti.

Pokud inicializovaný nebyl, provede se inicializace a zahraje jedna náhodná hra, ze které se poté uloží a vrátí výsledky.

Aby se stihlo v časovém limitu pro jeden tah více her, má náhodná hra omezenou hloubku (v závislosti na velikosti a stavu herní plochy). Probíhá poté tak, že nejprve zahrají všichni pacmani:

- s pravděpodobností 10 % se pokusí utéct od nejbližšího ducha,
- s pravděpodobností 10 % zahrají náhodně,
- s pravděpodobností zbylých 80 % udělají tah směrem k nejbližší kuličce.

Po pacmanech zahrají duchové - jsou-li v blízkosti pacmana tak hladově, pokud ne tak náhodně.

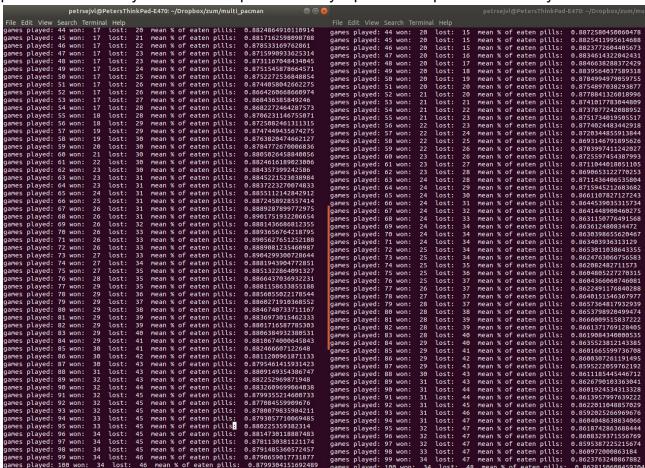
Výsledky náhodné hry se pak vyhodnotí a uloží.

Po doběhnutí časového limitu zahrají pacmani nejslibnější tah a duchové je doplní dle svých pravidel.

5. Implementace.

Pro implementaci jsem zvolil programovací jazyk Python. Pro grafické rozhraní jsem využil knihovny pygame.

Řešení jsem testoval na herní ploše 13x13 s 5 pacmany a 5 duchy. V rámci testování proběhlo 2 x 100 her, z toho pacmani vyhráli 68, prohráli 94 (zbytek her byl ukončen po zahrání 300 tahů). Průměrně v těchto hrách pacmani získali asi 87 % kuliček. Prohrané a nedokončené hry jsou důsledkem předeveším neoptimálního chování pacmanů v pozdních fázích hry. Směrodatná pro mě ale byla především procenta získaných kuliček.



6. Reference

Jak inspiraci jsem využil sérii videí rozebírající Monte carlo tree search algoritmus: https://www.youtube.com/watch?v=onBYsen2_eA