Theory of Computer Games 2022 – Project 1

311551069 余忠旻

Develop a simple player based on simple heuristics

我試了兩種 heuristics 方法

第一種，我按照現在 board 計算往上下左右四種方向做 slide 所得到的 reward，

對能取得最好的 reward 的方向來做實際的 slide

```
class reward_player : public agent {
public:
	reward_player(const std::string& args = "") : agent(args),
		opcode({ 0, 1, 2, 3 }) {}

	virtual action take_action(const board& before) {
		//std::shuffle(opcode.begin(), opcode.end(), engine);
		int bestOP = -1;
		board::reward bestReward = -1;
		for (int op : opcode) {
			board::reward reward = board(before).slide(op);
			if (reward > bestReward){
				bestReward = reward;
				bestOP = op;
			}
		}
		if (bestOP != -1) {
			return action::slide(bestOP);
		}
		else {
			return action();
		}
	}

private:
	std::array<int, 4> opcode;
};
```

第二種，在時間限制允許下，

我按照現在 board 計算往上下左右四種方向連續做兩次 slide 所得到的加總 reward

對能取得最好的加總 reward 的第一次 slide 方向來做實際的 slide

```
class twoSteps_player : public agent {
public:
	twoSteps_player(const std::string& args = "") : agent(args),
		opcode({ 0, 1, 2, 3 }) {}

	virtual action take_action(const board& before) {
		//std::shuffle(opcode.begin(), opcode.end(), engine);
		int bestOP = -1;
		board::reward bestReward = -1;

		for (int op1 : opcode) {
			board next1 = board(before);
			board::reward reward1 = next1.slide(op1);
			for (int op2 : opcode) {
				board next2 = board(next1);
				board::reward reward2 = next2.slide(op2);
				if (reward1 + reward2 >= bestReward) {
					bestOP = op1;
					bestReward = reward1 + reward2;
				}
			}
		}

		if (bestOP != -1) {
			return  action::slide(bestOP);
		}
		else {
			return action();
		}
	}

private:
	std::array<int, 4> opcode;
};
```