Simulácia šermu v 3D za pomoci UI
diplomová práca

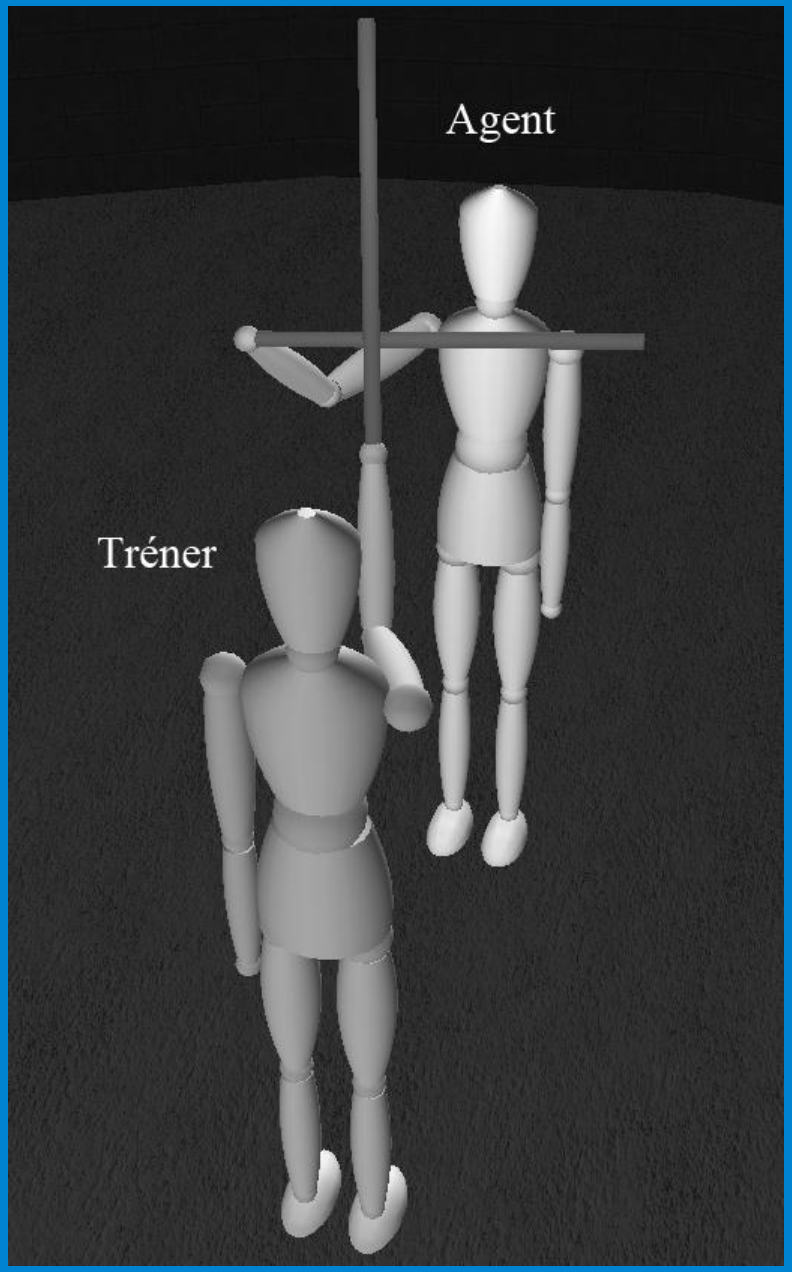
Autor: Bc. Jaroslav Blanář

Školiteľ: doc. Ing. Igor Farkaš, PhD.

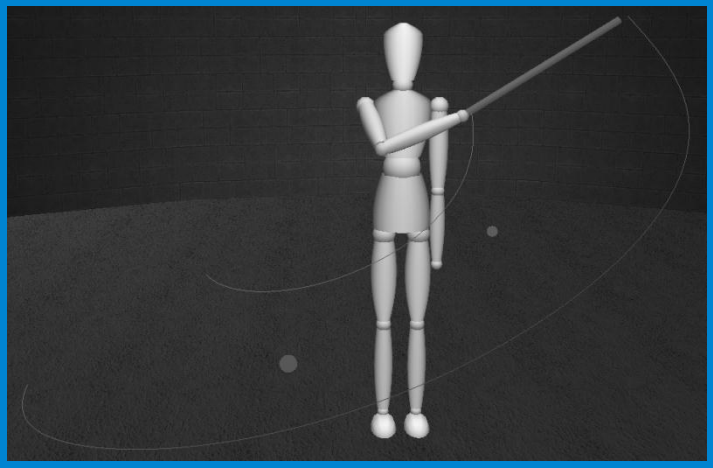
Cieľ

- Navrhnúť a implementovať adaptívneho agenta s využitím umelej inteligencie pre simuláciu šermovania.
- Úlohou agenta je naučiť sa obranné manévre a úspešne tak odrážať protivníkove výpady.
- Implementovať simulátor šermovania v už existujúcom virtuálnom prostredí alebo navrhnúť a implementovať vlastné virtuálne prostredie pre simulátor v zjednodušenej podobe.

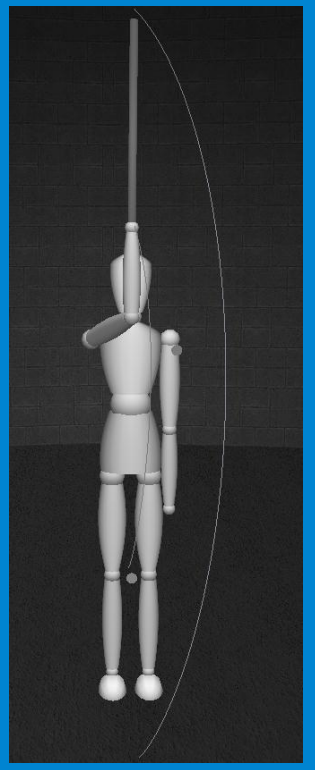
Modul šermovania



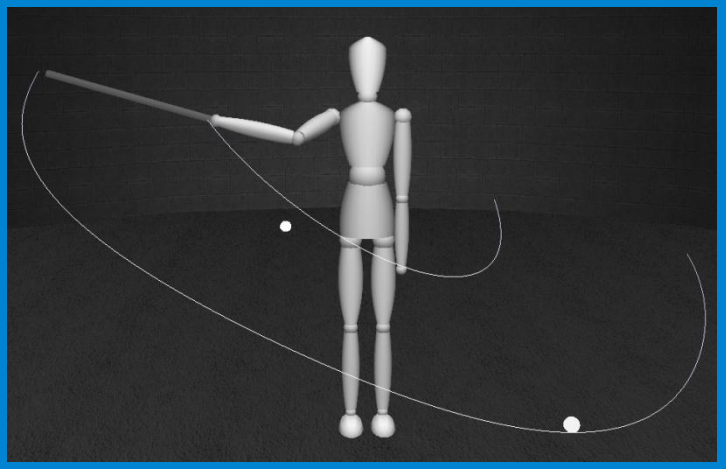
Typy výpadov



Výpad zľava

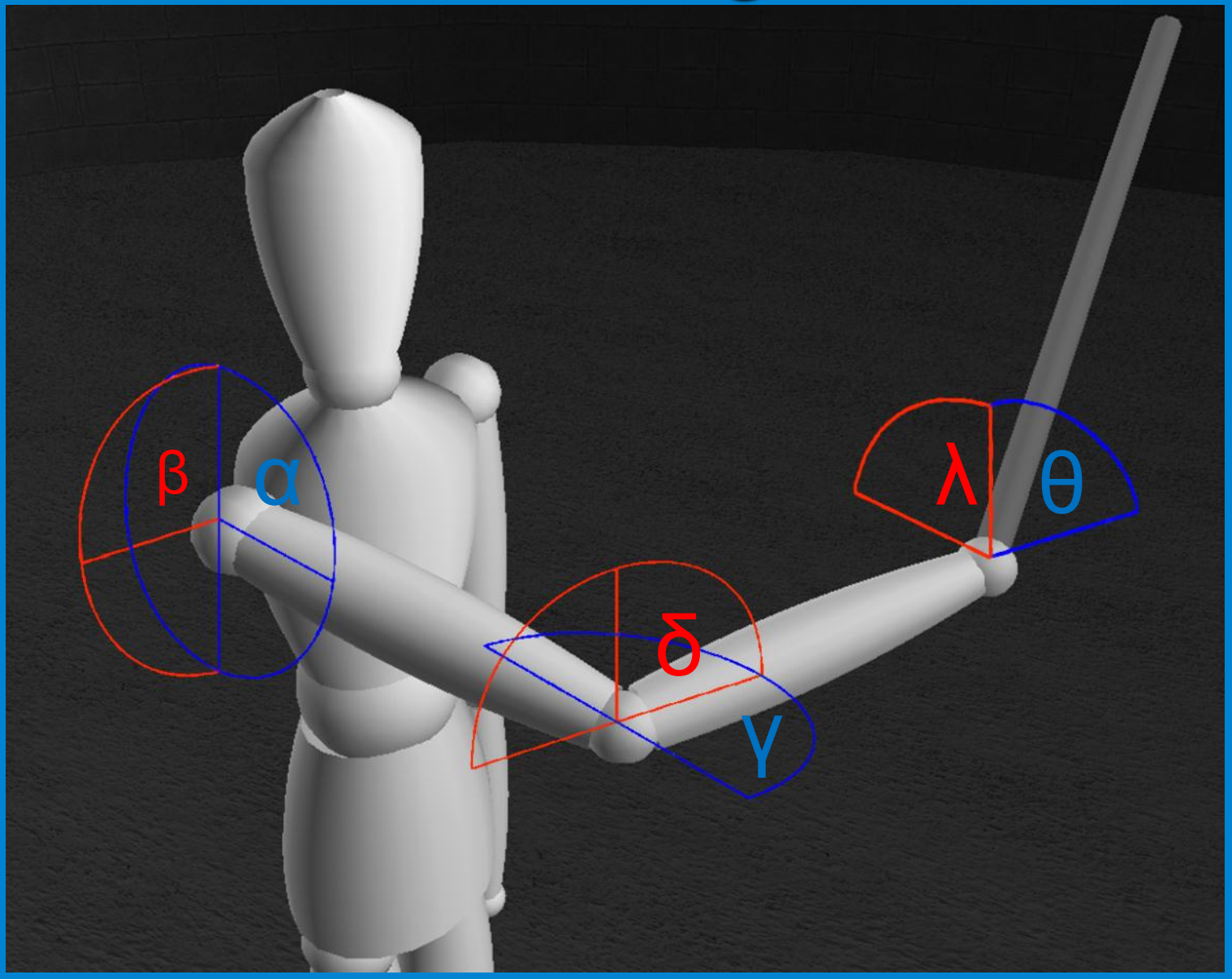


Výpad
zo stredu



Výpad sprava

Rameno agenta



Rameno agenta je ovládané pomocou 6 stupňov voľnosti.

Popis problému

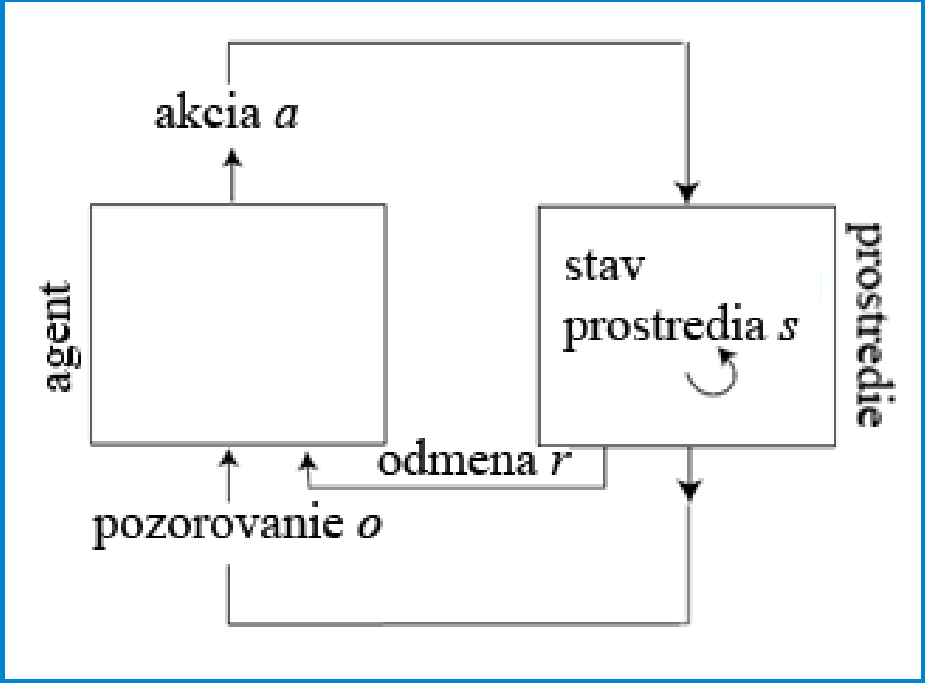
- Agentov aj trénerov meč je umiestnený v počiatočnej pozícii.
- Agent sa snaží voliť také akcie, ktorých aplikovaním sa jeho meč priblíži k trénerovmu meču.
- Cieľ je zablokovať trénerov meč, ktorý sa pohybuje smerom k agentovi.

Metódy UI

- Učenie s posilňovaním - Reinforcement learning
- Continuous Actor Critic Learning Automaton (CACLA)
- Neurónové siete

Učenie s posilňovaním

- s – konečná množina stavov
- a – konečná množina akcií
- r – funkcia, ktorá ohodnocuje správanie agenta
- o – informácie o aktuálnom stave pre agenta



CACLA

- ***Aktér***
  - svojimi akciami priamo zasahuje do prostredia.
  - v stave s_t musí zvoliť najvhodnejšiu akciu a_t,
  - počas trénovania sa snaží naučiť čo najlepšiu stratégiu.
  - reprezentovaný funkciou AC(stav), ktorá vráti najvhodnejšiu akciu na základe aktuálnej stratégie.

- ***Kritik***
  - nezasahuje svojou činnosťou do prostredia.
  - počas trénovania sa snaží čo najlepšie ohodnocovať kvalitu stavov.
  - reprezentovaný funkciou V(stav), ktorá ohodnotí daný stav.

Učiace pravidlo

- Použili sme doprednú neurónové siete, ktorých váhy sú parametre.
- Nech vektor θ^V je parametrom funkcie V, potom učiace pravidlo pre kritika je v tvare :

$$\theta^V_{i,t+1} = \theta^V_{i,t} + \alpha\delta_t \frac{\partial V_t(s_t)}{\partial \theta^V_{i,t}}$$

- Nech vektor θ^A je parametrom funkcie A, potom učiace pravidlo pre aktéra je v tvare :

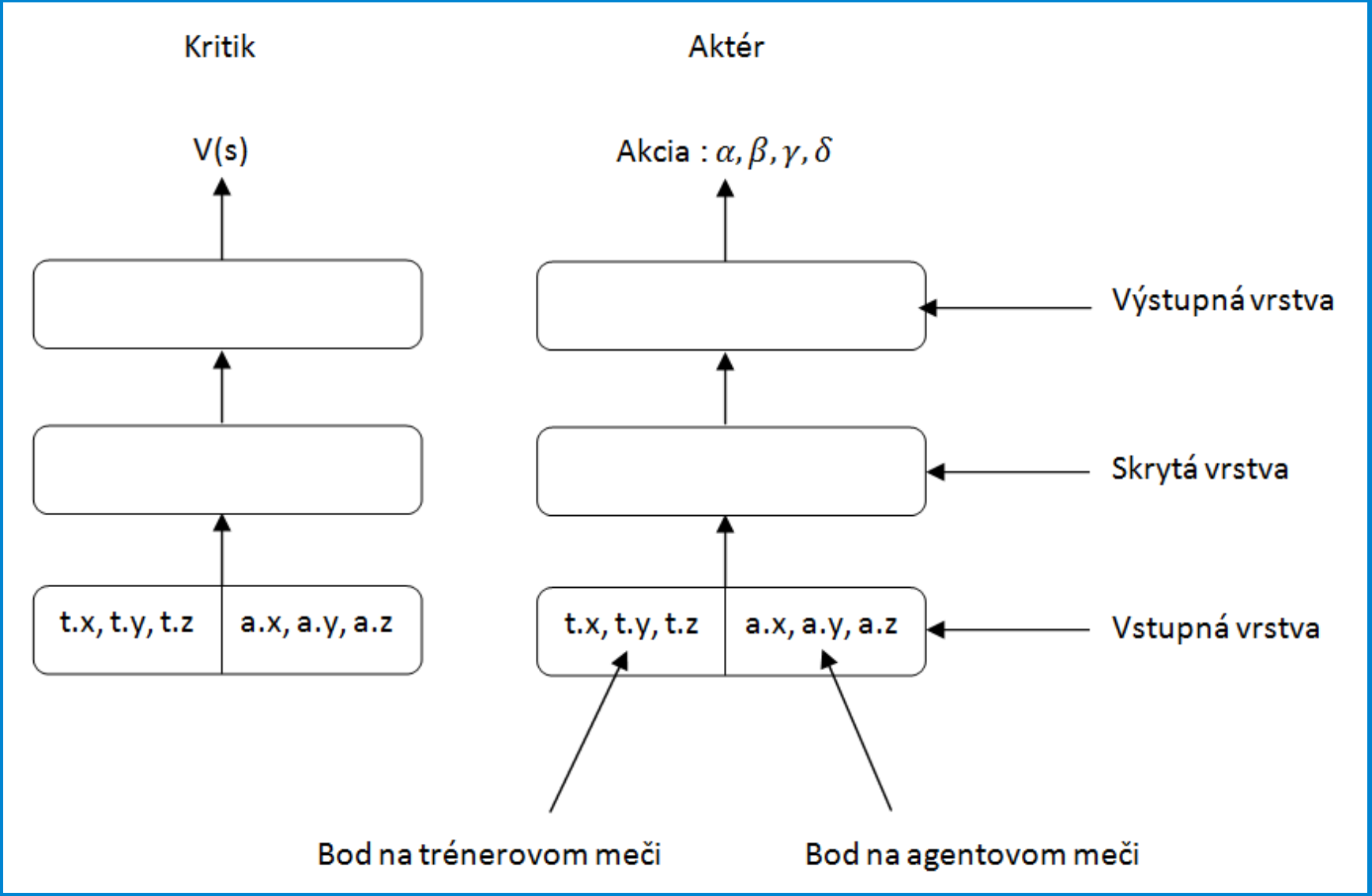
$$\text{if } \delta_t > 0 : \qquad \theta^A_{i,t+1} = \theta^A_{i,t} + \alpha\big(a_t - A_t(s_t)\big)\frac{\partial A_t(s_t)}{\partial \theta^A_{i,t}}$$

kde α je rýchlosť učenia a t je parameter času.

Algoritmus CACLA

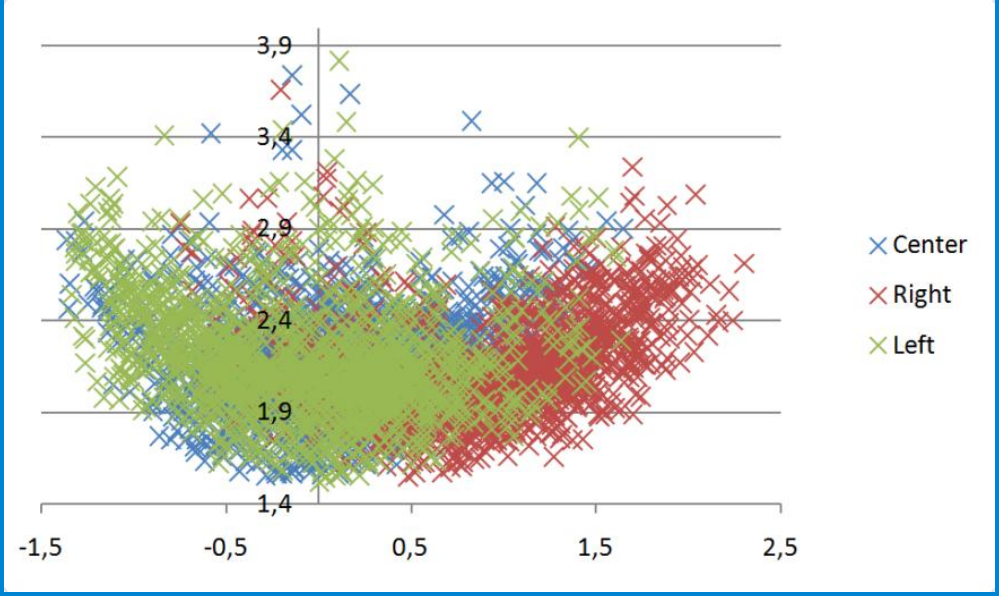
```
procedure Train(actualState) {
        actionActor = Actor.A(actualState);
        actionGausian = GetRandomAction(actionActor);
        newStateActor  = ApplyAction(actionGausian);
        rewardGausian = Reward(actionGausian);
        delta = rewardGausian + discountFactor * Critic.V(newStateActor) - Critic.V(actualState);
        if (delta  > 0){
                Actor.Update(actionGausian, actualState);
        }
        delta = delta + Critic.V(actualState);
        Critic.Update(detla,  actualState);
}
```

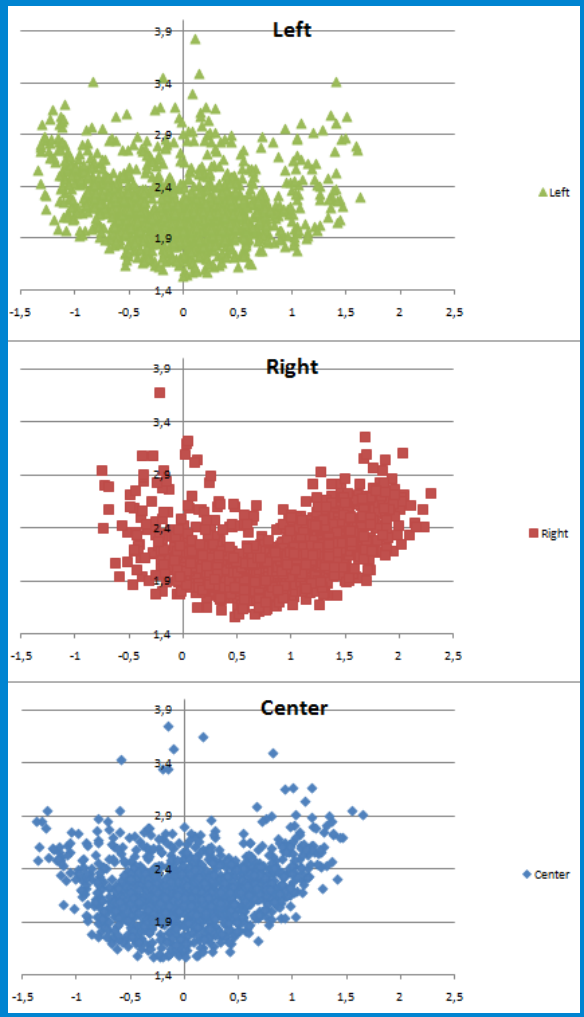
Návrh NS


Kritik
Aktér
V(s)
Akcia : $\alpha, \beta, \gamma, \delta$
Výstupná vrstva
Skrytá vrstva
Vstupná vrstva
t.x, t.y, t.z
a.x, a.y, a.z
t.x, t.y, t.z
a.x, a.y, a.z
Bod na trénerovom meči
Bod na agentovom meči

Problém s CACLA

- Algoritmus CACLA prepustí pri podmienke delta > 0 hodnoty s rozptylom v celom pásme, v dôsledku čoho sa agent nedokázal naučiť rozlišovať jednotlivé výpady a správne na ne reágovať.

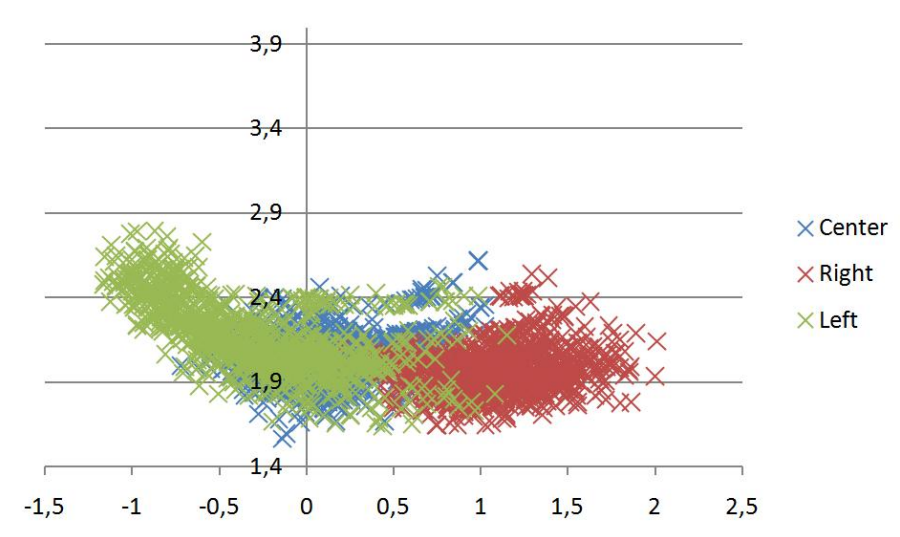


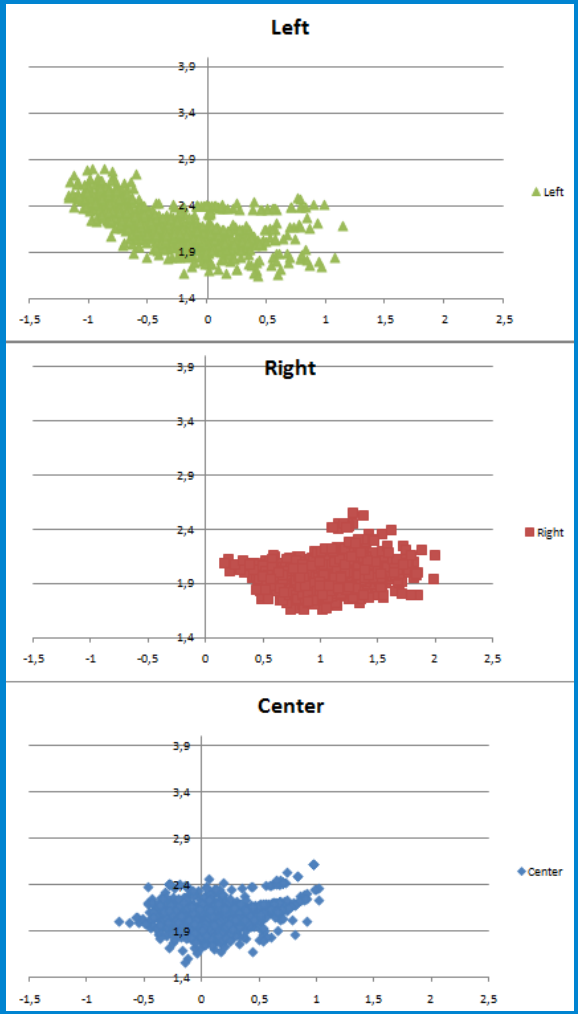


Modifikácia CACLA alg.

- Zmodifikoval som učiacu podmienku aktéra, ktorá viedla k zlepšeniu a schopnosti rozlíšiť jednotlivé výpady trénera.

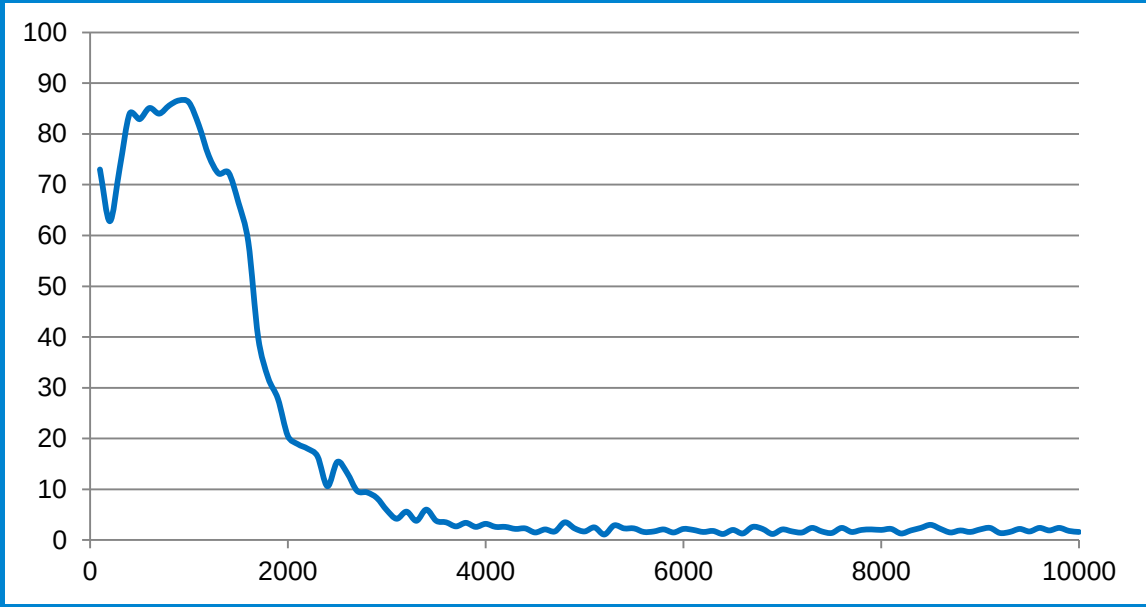
```
valueActorAction = Critic.V(newStateActor);
valueGausianAction = Critic.V(newStateGausian);
if (valueGausianAction > valueActorAction){
        Actor.Update(actionGausian, actualState);
}
```





Výsledky

- CACLA – sa dokázala naučiť efektívne len jeden druh výpadov.
- Modifikácia CACLA algoritmus – sa dokázala naučiť efektívne obraňovať proti viacerým druhom útokov, preukázala schopnosť rozpoznávať jednotlivé výpady. Dĺžka trénovania ~ 3000 tisíc epoch.



Priebeh chyby MCACLA počas 10 tisíc epoch.

Implementácia

- Visual C++
- OpenGL
- Beziérové krivky
- Windows

Ďakujem za pozornosť