Probabilitatea de a elimina un inamic ascuns în Counter-Strike

Introducere

Componența echipei:

- Cotivă David
- Piele Vlăduț-Ștefan
- Roșianu Radu-Daniel

Problema

În acest proiect, am ales să determinăm probabilitatea de a elimina un inamic nevăzut în **Counter-Strike** folosind focuri trase printr-un obstacol, fie el zid („wallbang”) sau o fumigenă. Problema este formulată ca o **simulare Monte Carlo** pentru a estima dacă un număr limitat de gloanțe (10, 15, 20) poate acumula suficient damage pentru a elimina un inamic static, ce este reprezentat printr-un model 2D.

Justificarea alegerii

Această problemă este relevantă din mai multe motive. Este insipirată dintr-un scenariu cât se poate de real întalnit în fiecare meci (dacă nu fiecare rundă!) de Counter-Strike. În timpul unei confruntări one-on-one, inamicul se ascunde repede în spatele obstacolului, lasându-ne stupefiați: ne irosim timp si gloanțe în speranța ca îl vom elimina sau nu? De acum încolo putem lua o decizie cu o riguroasă motivație matematică în spate.

Problema include elemente de jocuri de noroc (trageri aleatorii, pozitia necunoscută a inamicului) și integrează metode avansate de simulare si calcule geometrice pentru a determina probabilități. Desigur, pentru simplitatea proiectului, a calculul și a codului au fost făcute compromisuri. Se face abstracție de faptul ca inamic se poate mișca, își poate micșora modelul (crouch), jucătorul nu se va descurca să tragă gloanțele într-o distribuție uniformă (care să asigure rezultate asemănătoare simulării), iar **inamicul poate și riposta**!

Formularea matematică a problemei

Reprezentarea țintei:
Inamicul este reprezentat printr-un model geometric 2D care include zone specifice (cap, piept, mâini, stomac și picioare). Fiecare zonă are o formă și dimensiuni fixe:

- Cap: cerc de rază r_{head}.
- Piept: dreptunghi de dimensiuni $w_{\text{chest}} \times h_{\text{chest}}$.
- Mâini: dreptunghiuri verticale de dimensiuni $w_{\text{arm}} \times h_{\text{arm}}$.
- Stomac: dreptunghi de dimensiuni $w_{\text{stomach}} \times h_{\text{stomach}}$.
- Picioare: dreptunghiuri verticale de dimensiuni $w_{\text{leg}} \times h_{\text{leg}}$.

Poziția gloanțelor:
Gloanțele sunt generate aleatoriu în interiorul unei zone de tragere ($x \in [-1.5, 1.5], y \in [-1.5, 1.5]$). Să ne imaginăm că acestea sunt proporțiile unei cutii de pe hărțile de Counter-Strike.

Loviturile și damage-ul:
Fiecare glonț este verificat pentru a determina dacă lovește o zonă a inamicului:

- Dacă $\text{glonțul} \in \text{zonă}$, se aplică damage-ul specific zonei respective. Zonele sunt fidele hitbox-ului oficial din Counter-Strike, precum sunt și valorile de damage.
- Damage-ul total D_{total} este suma damage-ului acumulat de toate gloanțele.

Probabilitatea de eliminare:
Evenimentul de succes este definit ca $D_{\text{total}} \geq 100$ (viața inamicului).
Probabilitatea de succes este estimată ca:

$$P(\text{eliminare}) = \frac{\text{Număr de simulări reușite}}{\text{Număr total de simulări}}$$

5. **Metoda Monte Carlo**:
   - Generăm N simulări independente.
   - În fiecare simulare, generăm k gloanțe și calculăm D_{total}.
   - Aproximăm probabilitatea folosind:

$$P(\text{eliminare}) \approx \frac{\sum_{i=1}^{N} I(D_{\text{total},i} \geq 100)}{N}$$

   unde I este funcția indicator.

Algoritmul Monte Carlo utilizat

Descriere generală

Metoda Monte Carlo este utilizată pentru a estima probabilitatea ca un număr de gloanțe trase aleatoriu să elimine un inamic. În acest scop, am implementat un algoritm care generează pozițiile de impact ale focurilor trase de către jucator (static și el, precum inamicul). Determinăm daca glonțul lovește una dintre zonele hitbox-ului (cap/piept/mână/stomac/picior). În caz afirmativ, reducem health points-urile inamic-ului cu valoarea specifică pentru partea hitbox-ului lovită. Repetăm simularea pentru un număr suficient de mare pentru marja de eroare și nivelul de încredere dorit, calculând probabilitatea ca inamicul să rămână fără viață (damage-ul total să depășească 100).

Pași detaliați ai algoritmului Monte Carlo

1. Reprezentarea inamicului

Inamicul este reprezentat de un model geometric bidimensional, împărțit în următoarele zone:

- **Capul**: Cerc cu raza r_{head} centrat la $(0, 1.28)$.
- **Pieptul**: Dreptunghi centrat orizontal la $y = 0.8$, cu dimensiuni $w_{\text{chest}} \times h_{\text{chest}}$.
- **Mâinile**: Două dreptunghiuri verticale plasate de-o parte și de alta a pieptului, cu dimensiuni $w_{\text{arm}} \times h_{\text{arm}}$.
- **Stomacul**: Dreptunghi plasat imediat sub piept, cu dimensiuni $w_{\text{stomach}} \times h_{\text{stomach}}$.
- **Picioarele**: Două dreptunghiuri verticale sub stomac, fiecare cu dimensiuni $w_{\text{leg}} \times h_{\text{leg}}$.

2. Generarea pozițiilor gloanțelor

Pentru fiecare simulare:

- Generăm pozițiile gloanțelor aleatoriu în cadrul unei zone bidimensionale $x \in [-1.5, 1.5]$ și $y \in [-1.5, 1.5]$ utilizând distribuția uniformă:

$$x_{\text{glonț}} \sim \mathcal{U}(-1.5, 1.5), \quad y_{\text{glonț}} \sim \mathcal{U}(-1.5, 1.5).$$

3. Determinarea zonei lovite

Pentru fiecare glonț generat, verificăm dacă poziția acestuia intersectează una dintre zonele inamicului:

- Verificăm **capul** folosind ecuația cercului:

$$\sqrt{(x_{\text{glonț}} - x_{\text{cap}})^2 + (y_{\text{glonț}} - y_{\text{cap}})^2} \leq r_{\text{head}}.$$

- Verificăm **pieptul**, **mâinile**, **stomacul** și **picioarele** folosind inegalități care descriu limitele fiecărui dreptunghi:

$$x_{\text{glonț}} \in [x_{\min}, x_{\max}], \quad y_{\text{glonț}} \in [y_{\min}, y_{\max}].$$

- Dacă glonțul nu intersectează nicio zonă, considerăm că a ratat.

4. Calculul damage-ului

Dacă un glonț lovește o zonă, adăugăm damage-ul asociat acelei zone la totalul simulării:

$$D_{\text{total}} = \sum_{\text{focuri lovite}} D_{\text{zonă}},$$

unde $D_{\text{zonă}}$ este damage-ul asociat fiecărei zone:

- **Cap**: 135 HP
- **Piept**: 35 HP
- **Mâini**: 35 HP
- **Stomac**: 44 HP
- **Picioare**: 26 HP

5. Determinarea succesului simulării

O simulare este considerată reușită dacă damage-ul total acumulat depășește sau este egal cu pragul de eliminare (100 HP):

$$\text{Succes} = \begin{cases} 1, & \text{dacă } D_{\text{total}} \geq 100, \\ 0, & \text{altfel.} \end{cases}$$

6. Estimarea probabilității

După efectuarea N simulări, probabilitatea de eliminare este calculată ca proporția simulărilor reușite:

$$P(\text{eliminare}) = \frac{\text{Număr de simulări reușite}}{N}.$$

Utilizarea simulării pentru aproximarea soluției

Iterații multiple: Prin generarea unui număr mare de simulări (N), reducerea erorii statistice este garantată conform legii numerelor mari.
Convergență: Rezultatele obținute converg către probabilitatea reală de succes pe măsură ce numărul simulărilor crește.
Rezultate vizuale: Graficele generate oferă o perspectivă vizuală detaliată asupra rezultatelor simulării. Heatmap-ul și distribuția loviturilor pe zone vizualizează densitatea loviturilor și eficiența pe fiecare parte a corpului. Graficul performanței în funcție de numărul de gloanțe și cel al convergenței probabilității ilustrează creșterea șanselor de succes.

Prin acest algoritm, probabilitatea de eliminare poate fi estimată eficient și poate fi vizualizată în mod intuitiv. Este permisă explorarea parametrilor, precum numărul de gloanțe sau chiar modificarea damage-ului specific unei părți din hitbox.

Estimarea numărului minim de simulări necesare

Pentru a garanta o anumită marjă de eroare ϵ și un nivel de încredere $1 - \alpha$, putem utiliza **Teorema Limită Centrală (TLC)** pentru a estima numărul minim de simulări necesare.

Formula bazată pe TLC

Numărul minim de simulări este dat de formula:

$$N \geq \frac{z_{1-\alpha/2}^2 \cdot P(1-P)}{\epsilon^2}$$

Unde:

- $z_{1-\alpha/2}$: Quantilul pentru nivelul de încredere (ex. pentru 95%, $z_{1-\alpha/2} \approx 1.96$).
- P: Probabilitatea estimată (poate fi 0.5 pentru un caz conservator).
- ϵ: Marja de eroare dorită.

Cazuri de calcul

Deoarece probabilitatea de a elimina inamicul este greu de ghicit sau estimat și variază în funcție de numărul de focuri, am ales să calculăm numărul minim de iterații pentru multiple cazuri.

Caz conservator ($P = 0.5$)

Pentru $P = 0.5$, care maximizează $P(1 - P)$, nivel de încredere $1 - \alpha = 95\%$ ($z_{1-\alpha/2} = 1.96$), și o marjă de eroare $\epsilon = 0.01$:

$$N \geq \frac{1.96^2 \cdot 0.5 \cdot (1 - 0.5)}{0.01^2}$$

$$N \geq \frac{3.8416 \cdot 0.25}{0.0001}$$

$$N \geq 9604$$

Astfel, pentru un caz conservator, numărul minim de simulări este **9604**.

Caz pentru 10 gloanțe ($P \approx 0.15$)

Dacă probabilitatea estimată este $P = 0.15$, calculul devine:

$$N \geq \frac{1.96^2 \cdot 0.15 \cdot (1 - 0.15)}{0.01^2}$$

$$N \geq \frac{3.8416 \cdot 0.1275}{0.0001}$$

$$N \geq 4893$$

Astfel, pentru o estimare $P \approx 0.15$, numărul minim de simulări necesare este **4893**.

Caz pentru 15 gloanțe ($P = 0.20$)

Dacă probabilitatea estimată este $P = 0.20$, calculul devine:

$$N \geq \frac{1.96^2 \cdot 0.2 \cdot (1 - 0.2)}{0.01^2}$$

$$N \geq \frac{3.8416 \cdot 0.16}{0.0001}$$

$$N \geq 6147$$

Astfel, pentru o estimare $P = 0.20$, numărul minim de simulări necesare este **6147**.

Caz pentru 20 gloanțe ($P = 0.33$)

Dacă probabilitatea estimată este $P = 0.33$, calculul devine:

$$N \geq \frac{1.96^2 \cdot 0.33 \cdot (1 - 0.33)}{0.01^2}$$

$$N \geq \frac{3.8416 \cdot 0.2211}{0.0001}$$

$$N \geq 8496$$

Astfel, pentru o estimare $P = 0.33$, numărul minim de simulări necesare este **8496**.

Cu cât numărul de focuri trase se apropie de 30, cu atât converge și probabilitatea spre $P = 0.5$. Așa că vom rula un număr rotund de **10000** de simulări pentru Monte-Carlo. Aceste valori garantează că marja de eroare este $\epsilon = 0.01$ cu un nivel de încredere $1 - \alpha = 95\%$, indiferent de numărul de focuri.

Codul proiectului

Codul implementează o simulare Monte Carlo pentru a estima probabilitatea de a elimina un inamic folosind un număr dat de gloanțe (input de la utilizator) într-un spațiu bidimensional. Modelul inamicului este reprezentat geometric, iar fiecare zonă a corpului are un damage specific asociat. Simularea utilizează metode statistice pentru a genera poziții aleatorii ale gloanțelor și determină dacă acestea lovesc una dintre zonele inamicului. Rezultatele sunt apoi analizate prin grafice care vizualizează distribuția damage-ului, densitatea loviturilor și convergența probabilităților.

Structura generală a codului

Definirea parametrilor inamicului:

- Dimensiunile fiecărei zone (cap, piept, mâini, stomac, picioare).
- Damage-ul asociat fiecărei zone.
- Zona de tragere ($[-1.5, 1.5]$).

Funcții principale:

- `draw_enemy` : Desenează grafic modelul inamicului pe plan 2D.
- `check_hit` : Determină zona lovită de fiecare glonț.

- `monte_carlo_simulation` : Realizează simulările Monte Carlo pentru a calcula probabilitatea de eliminare.
- `get_theoretical_probability` : Returnează probabilitatea teoretică pentru un anumit număr de gloanțe.
- `analyze_errors` : Analizează erorile și calculează intervalul de încredere.
- Funcțiile de graficare (`plot_damage_distribution` , `plot_hits_per_zone` , `plot_heatmap` , etc.).

Execuția codului:

- Se desenează inamicul.
- Se rulează simularea Monte Carlo pentru un număr specific de gloanțe și simulări.
- Se analizează erorile și se generează grafice.

Explicația funcțiilor folosite

`draw_enemy`

Această funcție creează o reprezentare vizuală a modelului inamicului pe un plan 2D:

- Capul este reprezentat printr-un cerc, iar restul zonelor sunt dreptunghiuri.
- Dimensiunile fiecărei părți sunt definite prin variabile globale.
- Funcția folosește biblioteca `matplotlib` pentru a afișa graficul.

`check_hit`

Determină dacă un glonț lovește o zonă a inamicului:

- Calculează dacă glonțul se află în limitele geometrice ale fiecărei zone.
- Returnează numele zonei lovite (e.g., „head”, „chest”) sau `None` dacă glonțul ratează.

`monte_carlo_simulation`

Efectuează simularea Monte Carlo:

- Pentru fiecare simulare, generează un număr specific de gloanțe cu coordonate aleatoare.
- Utilizează `check_hit` pentru a determina damage-ul total acumulat.
- Numără simulările în care damage-ul depășește 100 și calculează probabilitatea finală.

`get_theoretical_probability`

Returnează probabilități teoretice predefinite pentru un număr dat de gloanțe.

`analyze_errors`

Calculează erorile absolute, relative și intervalul de încredere pentru probabilitatea estimată:

- Utilizează formula erorii standard și distribuția normală pentru calcul.

Funcțiile de grafing

- **`plot_damage_distribution`** : Afișează o histogramă care arată frecvența damage-ului total.
- **`plot_hits_per_zone`** : Reprezintă numărul de lovituri pe fiecare zonă a corpului.
- **`plot_heatmap`** : Creează un heatmap care arată densitatea loviturilor pe modelul țintei.
- **`plot_performance_by_shots`** : Arată probabilitatea de succes în funcție de numărul de gloanțe trase.
- **`plot_probability_convergence`** : Vizualizează convergența probabilității pe măsură ce cresc simulările.

De asemenea, pe GitHub este uploadată într-un fișier separata funcția folosita pentru creația GIF-urilor folosite la prezentare.

Modul de apelare și execuție

Execuția are loc în ordinea următoare:

1. **Input utilizator**:
   - Utilizatorul introduce numărul de gloanțe (5, 10, 15, 20, 25).
2. **Desenarea modelului**:
   - `draw_enemy` este apelată la început pentru a afișa modelul grafic al inamicului.
3. **Simulare Monte Carlo**:
   - `monte_carlo_simulation` rulează simularea și returnează probabilitatea estimată, damage-ul acumulat și alte statistici.
4. **Analiza erorilor**:
   - `analyze_errors` compară probabilitatea estimată cu cea teoretică și afișează erorile.
5. **Generarea graficelor**:
   - Graficele sunt generate în următoarea ordine (unele dintre ele se încarcă mai greu):

a. Distribuția damage-ului.
b. Repartizarea loviturilor pe zone.
c. Heatmap-ul loviturilor.
d. Performanța în funcție de numărul de gloanțe.
e. Convergența probabilității.

Librarii necesare

Pentru ca aplicația să funcționeze este nevoie doar de 2 librării externe.

```
pip install numpy
pip install matplotlib
```

Execuție

Pentru execuție este suficient rulați in folder-ul proiectului comanda:

```
python ./main.py
```

Urmează să introduceți numărul de focuri trase. (alegeți dintre cele afișate pentru analiza corectă a erorilor)

```
Introduceți numărul de focuri trase (5/10/15/20/25)
15


Introduceți numărul de focuri trase (5/10/15/20/25)
25
```

In consolă va fi afișată probabilitatea de a ucide jucătorul inamic și analiza erorilor. Apoi se vor deschide graficele (modelul inamicului, convergența probabilităților șamd). Poate dura o perioadă de câteva secunde ca acestea să se încarce.

Bibliografie

https://en.wikipedia.org/wiki/Monte_Carlo_method

https://matplotlib.org/stable/contents.html

https://numpy.org/doc/stable/

https://en.wikipedia.org/wiki/Central_limit_theorem