Pszeudo-random tesztmanőver generálás robusztussági tesztekhez

Mikes Marcell

# Bevezetés

## Feladat oka, célja

### Tesztelés fontossága

A fejlesztési folyamat fontos része a tesztelés, mert így győződhetünk meg arról, hogy egy átadott termék úgy működik, ahogy azt a megrendelő szeretné. A szoftverfejlesztés során bonyolult rendszereknél körültekintés ellenére akaratlanul is kerülhet hiba a szoftverbe. A cél, hogy ezeket a hibákat még időben beazonosítsuk és kijavítsuk. A thyssenkrupp által fejlesztett termékek biztonságkritikus rendszerek, így kiemelt figyelmet kell fordítani a működésük helyességének biztosítására. A termékeknek továbbá meg kell felelniük az autóipari biztonsági szabványoknak is. Biztonság szempontjából az (ISO26262) és a minőség szempontjából pedig az ASPICE.

### Robusztussági tesztek jelenleg

Jelenleg a thyssenkrupp által használt robusztussági tesztek, egy játékszimulációban levezetett mérést játszanak le újra a tesztpadon. A robusztussági teszt futtatása közben pedig hibákat injektálhat a tesztmérnök. A különböző vezetési módokhoz több mérés is van, de összesen limitált mennyiségű, így a tesztesetek száma is limitált. Tesztelés szempontjából jobb lenne, ha bármennyi egymástól eltérő tesztmanővert lehetne futtatni, ezáltal növelve a tesztelt munkapontokat. Több munkapont tesztelése esetén pedig akár olyan hibát is találhatunk a rendszerben, amit a jelenlegi robusztussági teszttel nem vettünk volna észre.

### ?

Egy tesztmanőver készítéshez két bemeneti jelet kell definiálnunk. Az egyik a járműsebesség, a másik pedig a kormányszög. A kettő jel és a hozzájuk rendelt idő fogja megadni, hogy adott pillanatban hány fokon álljon a kormány és milyen sebességgel közlekedjen az autó.

# Feladat tervezése

## Véletlenszerű tesztmanőver készítés

A tesztmanőver elkészítését a korábban ismertetett Perlin-zaj algoritmussal fogom megvalósítani. Ez az algoritmus fogja a kormányszög és járműsebesség jelek alapját adni, amiket aztán módosítani kell a valósághű tesztmanőver elérése érdekében.

### Pszeudo-random generálás

A pszeudo-random szám generálás lényege, hogy statisztikailag véletlenszerűnek látszanak a kimenetei, de közben determinisztikus és megismételhető legyen a működése. Az ilyen algoritmusoknak egy kezdőállapotot kell megadni, amit „seed-nek” nevezünk. Azonos kezdőállapotra sorrendben és értékben is megegyező véletlen számokat fogunk kapni.

Tesztelésnél fontos, hogy a teszt procedúra reprodukálható legyen. Például azért, hogy egy hiba kijavítása után ellenőrizni lehessen újra a rendszer működését. A tesztmanőver elkészítéséhez tehát egy ilyen véletlenszám generátort, a „numpy” python könyvtárból fogok használni. Minden elkészült tesztesethez használt „seed” a tesztmanőverekkel együtt tárolva lesz, későbbi futtathatóság érdekében.

## Skálázás tartományba

A Perlin-zaj által generált értékek értékkészletét korábban már meghatároztam. Az így kapott kormányszöget és járműsebességet valósághű határértékek alapján kell felskálázni. A kormányszögnél ez a maximum kormányszög lesz, amit az EndPos határoz meg. A járműsebességnél a tartomány a kért vezetési mód alapján kerül beállításra. Mindkét esetben a tesztmérnök adhatja meg a kívánt határértékeket.

## Eloszlás változtatása

A Perlin-zaj eloszlása egy normális eloszláshoz hasonlít, ahol az értékek középen csúcsosodnak. Az értékkészletéből és a normális eloszlásra hasonlításából következik, hogy az átlag, vagyis az eloszlás központi része mindig 0-hoz közelít.

A képen diagram, szöveg, képernyőkép, Diagram látható

Automatikusan generált leírás

Perlin-zaj eloszlása

Ez ugyan a kormányszögnél egy jó tulajdonságnak mondható, mert a középpont ennél is 0 és minél messzebb van a középponttól annál valószínűtlenebb az érték megjelenése, de a járműsebességhez már nem ideális. A különböző vezetési módoknál normális eloszlással nem lehet elérni a kívánt működést. Autópályás vezetésnél nem valósághű, hogy egy nagyobb és kisebb, a középpontól ugyanannyira lévő sebesség ugyanazt a valószínűséget kapja. [\*Ha van, statisztika link ezzel kapcsolatban\*] Például autópályán nem feltétlenül egyenlő a 120 km/h és 80 km/h járműsebesség gyakoriságának a valószínűsége. A valósághoz közelebbi működés érdekében ezért, a Perlin-zajhoz tetszőleges eloszlást kell rendelni. A tesztmérnök így meghatározhatja, hogy az adott teszt milyen vezetési móddal fusson, milyen gyakran forduljanak elő adott értékek.

## Kormányszög és járműsebesség kapcsolata

Ha a kormányszöget és járműsebességet végig külön kezelnénk akkor a végén olyan eredményeket kapnánk, amelyek bár külön-külön helyesek lennének, együtt nem fednének le egy élethű autóvezetést. Ezért össze kell kapcsolni a kettőt, hogy legyen hatásuk egymásra és a tesztmanőver összeségében realisztikus legyen.

Az összekapcsoláshoz egy tanulmányt [link] vettem alapul, ahol a kormányszöget mérték, adott járműsebességeknél városi közlekedés során. A mérések alapján egy limitet állítottak be a kormányszögre, ami alatt az autóvezető még kontrollálni tudja az autót. Ezt a limitet aztán egy tesztpályán verifikálták.

A járműsebességet vettem alapul és ahhoz igazítottam a kormányszöget. A kormányszög limitet tehát a tanulmány mérései alapján állítottam be, ezzel garantálva az algoritmus, a valós vezetéshez minél közelebbi működést.

A képen szöveg, képernyőkép, sor, Diagram látható

Automatikusan generált leírás

kormányszög limitálás járműsebesség alapján

## Kormányszög és járműsebesség szabályozása

Fontos, a gyorsulás és a kormány eltekeredésének sebessége is szabályozva legyen. A kormányszög és járműsebesség felskálázása után előfordulhat olyan, hogy túl nagy a két egymást követő érték közötti különbség. Túl nagy különbség esetén pedig eltérne egy valósághű autóvezetéstől és legrosszabb esetben kárt is tehetne a tesztpadokban. Ezért mindenképpen szükség van egy olyan függvényre is, ami tetszőleges szögsebességre és gyorsulásra tud szabályozni.

# Megvalósítás

## Folyamatábra

\*aktivitás diagram az algoritmusról\*

## Perlin-zaj

\*vsp 1 octávos Perlin-noise kép\*

### esetleg 1D vs 2D

\*kép az 1D Perlin-noise\*

### Octávok

A Perlin-zaj algoritmus megvalósítása során implementáltam az úgynevezett „octave-okat” is. Az „octave” több különböző frekvenciájú Perlin-zajok összeadása. Leggyakoribb megoldás, amit én is használtam, hogy minden „octave” frekvenciája az előzőének a duplája. Ennek segítségével el tudunk érni egy sokkal részletesebb zajt. Tesztmanőver esetében pedig élethűbb kormányzást és sebességváltozást kaphatunk vele.

Ideális „octave” meghatározása függ a vezetési mód, bemeneti paraméterek és a teszt hosszúsággától is, ezért a tesztmérnök maga határozhatja meg milyen részletes zajt szeretne használni.

\*vsp több octávos Perlin-noise kép (esetleg side by side 1 octávossal, lehet az túl nagy lenne)\*

## Eloszlás változtatása

Az eloszlás tetszőleges beállításához két olyan jelre van szükség, amelyek más tartományban vannak. Ha az egyik jelből időközben átváltunk a másikba akkor elérhetünk egy normális eloszlástól eltérő jelet. Például, ha van egy jelünk 40 és 60 között és egy másik 90 és 130 között, akkor az ezekből készült jel eloszlását úgy szabhatjuk tetszésünkre, hogy az elsőnek és a másodiknak is csak egy részét használjuk fel. És az, hogy mennyi részét használjuk fel a jeleknek meghatározza, hogy milyen eloszlású lesz az elkészült jelünk.

A tetszőleges eloszlás megvalósításához a jelenlegi járműsebességtől eltérő Perlin-zajt kell generálnunk. Ezt a két jelet illesztem majd össze az eloszlás megváltoztatásához. A tesztmérnöknek egy olyan „dictionary-t” kell megadnia, amiben benne vannak a két járműsebesség határértékei és az, hogy milyen arányban szerepeljenek az elkészült jelben. A két jelet először a megadott határértékhez tolom, illetve aszerint skálázom. Aztán az egyikből másikba való átmenetet egy 0-tól 1-ig feltöltött tömb segítségével valósítom meg. A bemenetként megkapott arány szerint végig megyek a tömbön és az elsőt megszorzom az egyből kivont éppen aktuális elemmel, a másodikat pedig az aktuális elemmel szorzom, aztán összeadom az értéküket. Ezáltal a két jelet összetudom úgy kötni, hogy ne legyen hirtelen értékváltozás az összecsatolás helyén. A végeredmény egy olyan járműsebesség, amit a tesztmérnök bemenetei alapján tetszőleges eloszlással és határértékekkel hoztam létre.

A képen szöveg, képernyőkép, diagram, Diagram látható

Automatikusan generált leírás

Módosított Perlin-zaj eloszlása

## Kormányszög limitálás

\*stwa kormányszög rate limit kép\*

# Kiértékelés

## Végső kinézet, futatott mérések

## Összehasonlítás jelenlegi robusztussági tesztekkel

## Következtetések levonása