DIPLOMAMUNKA

Uzonyi Béla

Debrecen

2008

Debreceni Egyetem
Informatikai Kar

MOZGÁSSZIMULÁCIÓ

Témavezető:
Dr. Herendi Tamás
Egyetemi adjunktus

Készítette:
Uzonyi Béla
Programtervező matematikus

Debrecen
2008

Tartalomjegyzék

1 Bevezetés

Napjainkra a számítógépes szimuláció a számítástechnika egyik legnépszerűbb és leglátványosabb területévé vált annak köszönhetően, hogy a mindenki számára elérhető asztali számítógépek teljesítménye elérte a futtatásukhoz szükséges szintet. Ez lehetővé tette a járműdinamika valós idejű szimulálását anélkül, hogy a programozóknak bitről bitre optimalizált kódot kellene írniuk. A mai grafikus hardverek pedig kellő szépségű és részletességű virtuális világok megjelenítésére képesek megfelelő gyorsasággal. A szimulátorok általános hasznának bemutatására valószínűleg a vezetés szimulációja a legalkalmasabb: a szórakoztatás mellett vezetni tanulók alkalmazhatják oktatóeszközként, autógyártók és akár versenycsapatok használhatják tesztelésre, ezzel nagymértékben csökkentve a költségeket és a veszélyt. Természetesen ehhez elengedhetetlen, hogy a szimuláció valósághű legyen.

Manapság szintén gyorsan fejlődő terület az önműködő járművek technológiájáé. Ezen fejlesztéseknek az a célja, hogy a járművek emberi beavatkozás nélkül, az útjukba kerülő akadályokat kikerülve, biztonságosan eljussanak egy előzetesen megadott helyre. A navigáció során a fedélzeti számítógép a globális helymeghatározó rendszerre és a fedélzeti érzékelőkre (kamerák, radarok vagy akár lézeres letapogatók) hagyatkozik. Ilyen robotjárművek számára versenyeket is írnak ki, a legnevesebb ezek közül talán a DARPA Grand Challenge. Ezt először 2004-ben a Mojave sivatagban rendezték meg, ekkor azonban még egy jármű sem tudta teljesíteni a 240 km-es távot. A technológiai fejlődést jól mutatja, hogy a tavalyi évben már városi körülmények között versenyeztek a járművek, és a tizenegy indulóból hat elérte a 85 km-re lévő célt. Ezt a versenyt, melyen főként egyetemi csapatok vesznek részt, az amerikai katonai fejlesztési ügynökség (DARPA) rendezi, azonban az autógyártók számára sem közömbös.

Diplomamunkám témájául egy virtuális négykerekű jármű mozgásának szimulálását és automatikus vezérlésének programozását választottam. Egyik célom tehát az volt, hogy egy virtuális világban mozgó jármű viselkedését valósághűen szimuláljam, a másik pedig az, hogy ez a jármű felhasználói beavatkozás nélkül el tudjon navigálni az előre megadott koordinátákra.

A programozási nyelvet illetően a választásom a C++ nyelvre [1] [2] esett, mivel moduláris, hordozható és a valósidejű számítások elvégzéséhez megfelelő teljesítményt nyújtó kód írását teszi lehetővé. A megjelenítéshez az OpenGL grafikus könyvtárat [3] [4] [5] [6] használtam, mert rugalmas, hatékony és megfelelő mértékben platformfüggetlen. Használtam a GLUT (OpenGL Utility Toolkit) könyvtárat [7] is, mellyel több OpenGL-funkció könnyebben kezelhető, ugyanakkor az alkalmazás továbbra is független marad az ablakozó rendszertől.

Habár a jármű és a környezet, ahol az mozog, virtuális, nem feltételeztem, hogy a jármű minden időpillanatban ismeri környezetének minden részletét. Egy valóságos robotjárműhöz hasonlóan szenzorai csak bizonyos távolságra „látnak el”, így a terepviszonyokat csak ezen a távolságon belül érzékeli.

2 A szimulációs alkalmazás működése és felépítése

A szimuláció tulajdonképpen egy valós rendszer vagy folyamat utánzását jelenti. Amennyiben az elkészített fizikai vagy számítógépes modell kellően pontos, lehetővé válik az eredeti rendszer tanulmányozása a modellen keresztül. Szimuláció alkalmazását emellett több tényező is indokolttá teheti. Egyes vizsgálni kívánt rendszerek például csak nehezen figyelhetők meg. Más rendszerek működését csak részlegesen ismerjük, de előállhatunk elméletekkel ezzel kapcsolatban, amelyeket a szimuláció segítségével megerősíthetünk vagy megcáfolhatunk. Kíváncsiak lehetünk akár a rendszer jövőben várható viselkedésére is.

A mi esetünkben a modellezni kívánt rendszert egy jármű és annak fizikai környezete jelenti. Ezért az alkalmazásnak ismernie kell a jármű tulajdonságait, működését, viselkedését, illetve ezen túl magát a környezetet és a dinamika törvényeit. Ha ezeket pontosan definiáljuk, valósághű szimulációt kaphatunk.

A szimulációs alkalmazás működésének magját a szimulációs ciklus alkotja. Az alkalmazást elindítva a szükséges adatok betöltése és a szimuláció alaphelyzetbe állítása után elkezdődik az első szimulációs ciklus végrehajtása. Amint az befejeződött, kezdődik a másodiké, és így tovább, amíg a felhasználó nem kéri a megszakítását, vagy esetleg valamilyen végzetes hiba nem történik.

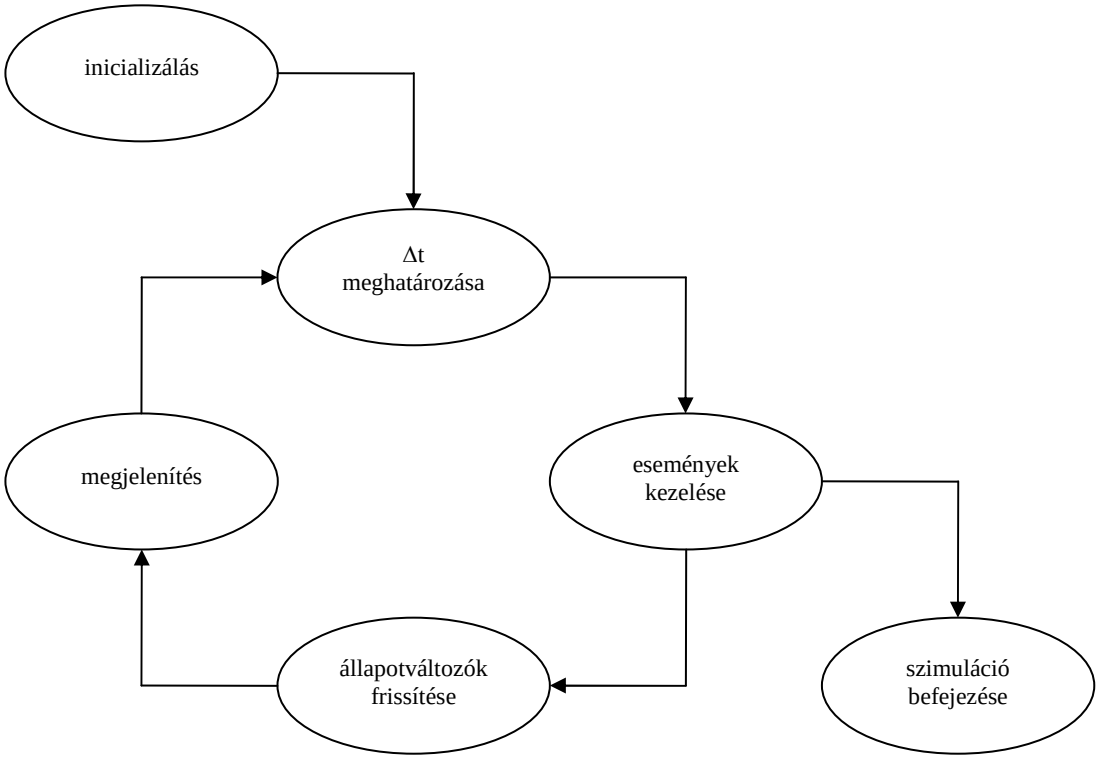
Mielőtt az első ciklust elindítanánk, inicializálni kell az összes állapotváltozót. A szimulációt leíró állapotváltozó például a jármű pozíciója, állásának iránya, sebessége, kormányának és pedáljainak állása. Meg kell határozni bizonyos egyéb paramétereket is, például a motor nyomatékát vagy azt, hogy maximálisan meddig lehet elfordítani a kormányt és ez az első kerekek hány fokos elfordulását eredményezi. Inicializálni kell továbbá a környezetet is, be kell tölteni a járművet és az egyéb objektumokat leíró modelleket és a kirajzoláshoz szükséges textúrákat. Ha a járművet nem a felhasználó irányítja, hanem automatikusan próbál eljutni egy meghatározott célhoz, akkor az előzetes pályaterv elkészítése is itt történik meg.

A ciklus első lépése az előző ciklus végrehajtása közben eltelt Δt idő meghatározása. Az első ciklus esetén ezt vehetjük nulla másodpercnek. Ezen idő ismerete elengedhetetlenül szükséges, mert a legtöbb állapotváltozót csak ennek függvényében tudjuk frissíteni.

Például a jármű új pozíciójának kiszámításához nem elég ismerni az aktuális pozícióját, sebességét és egyéb paramétereit, az időre is szükségünk van.

A következő lépés azoknak az eseményeknek a meghatározása és hatásának kezelése, amelyek az előző ciklus hasonló lépése után következtek be. Ez lehet például egy billentyű lenyomása, amellyel kamerát szeretnénk váltani, vagy a kormányt szeretnénk elfordítani, esetleg a gázpedált lenyomni, ha a felhasználó irányítja a járművet. Ha a felhasználó a szimuláció megszakítását kéri, akkor az aktuális ciklus megszakad, és a program befejeződik.

Ez után az állapotváltozók frissítése következik. Az első lépésben meghatározott Δt idő és az állapotváltozók aktuális értéke alapján kiszámítjuk az új értéküket. Például, amikor a felhasználó irányítja a járművet, és éppen lenyomva tartja a kormány jobbra forgatását előidéző billentyűt, továbbá ismerjük a kormány aktuális állását és azt, hogy egységnyi idő alatt hány fokkal tudjuk elfordítani azt, akkor kiszámítható az új kormányállás. Amikor a jármű automatizáltan működik, akkor az előzetes pályaterv és a javasolt működési paraméterek alapján fogja kezelni a kormányt és a pedálokat. Ilyen paraméter például a javasolt utazósebesség.



Az utolsó lépés a jármű, a környezet és az egyéb objektumok kirajzolása a frissített állapotváltozóknak megfelelően. A jármű tehát az új pozícióban az új kormányállással az új kamerapozícióból nézve fog megjelenni. Ahhoz, hogy a mozgás kellően folytonos hatást keltsen, másodpercenként legalább 60 ilyen kirajzolásnak kell történnie, azaz a Δt értékének 1/60 másodperc alatt kell lennie. Ez a követelmény egy átlagos mai számítógépnek nem okozhat problémát.

Tervezéskor fontosnak tartottam, hogy a szimuláció független legyen a tereptől és a járműtől is, ami lehetővé teszi, hogy a program a kód módosítása nélkül többféle terepen különböző járművek mozgásának szimulálására legyen képes. Ennek érdekében mind a terep, mind a jármű jellemzőit egy-egy konfigurációs fájlból olvassa be a program. Egy harmadik konfigurációs fájl pedig megadja az előző két állomány elérési útvonalát és néhány futási jellemzőt, melyek sem a terephez, sem a járműhöz nem köthetők. Ennek a fájlnak a nevét futási paraméterként is megadhatjuk.

A megvalósítás során törekedtem a modularitásra és az újrafelhasználhatóságra. Minden fontosabb objektumot, így a járművet és a terepet is a neki megfelelő osztály egy-egy példánya reprezentál. Ezen osztályok képesek elvégezni a szükséges adatok betöltését, a hozzájuk tartozó állapotváltozók frissítését és a reprezentált objektum kirajzolását is. Ezeket az objektumokat a szimulációs motort reprezentáló osztály fogja össze, melynek feladata a szimuláció vezérlése. A főprogramot reprezentáló osztály felelős az operációs rendszerrel és a felhasználóval való kapcsolattartásért.

3 A terep

Egy átlagos, vezetést szimuláló alkalmazásban a virtuális világ csupán magát a versenypályát és annak közvetlen környezetét ábrázolja. A virtuális valóság azonban olyan világot mutat be, ahol a felhasználó lehetőleg minden irányba szabadon mozoghat. Ezért szerencsésebb olyan megoldást választani, amely követi ezt az elvet. A terepmodellezés alkalmazása emiatt kézenfekvőnek tűnik.

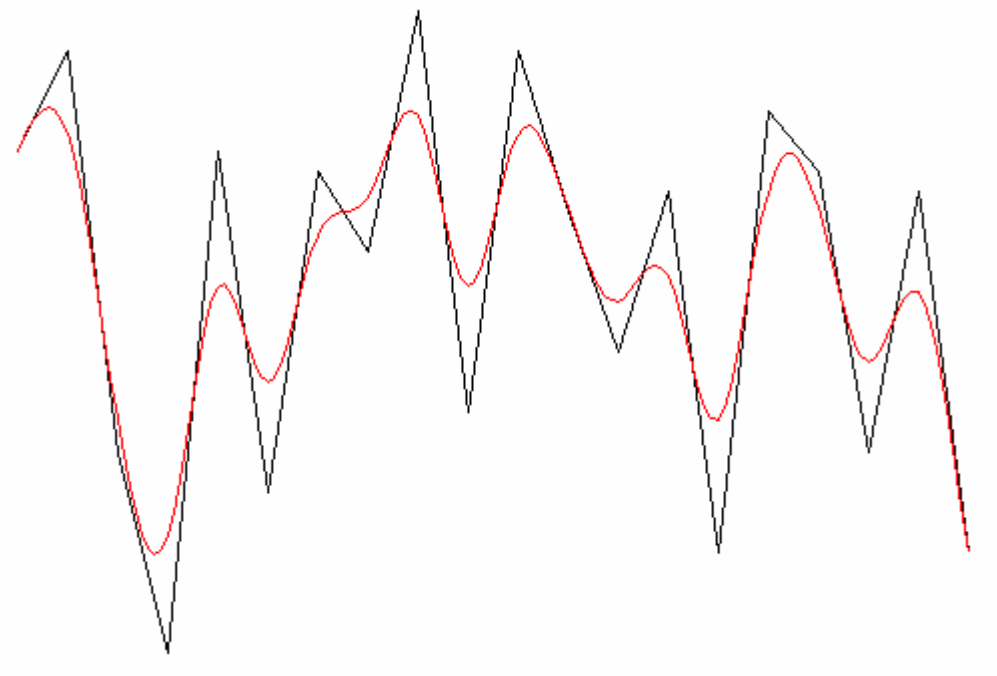
Önmagában a terepmodellezésről akár több tanulmányt is lehetne készíteni [8]. Bár közvetlen céljaim között nem szerepelt összetett terepmodell elkészítése, fontos belátni, hogy a szimulációhoz elengedhetetlen a megfelelő részletességű és simaságú terep megléte. Sőt, a járműdinamikai modell megadása mellett a szimuláció írásának másik fontos része a virtuális világ grafikus modelljének elkészítése. Hogy a szimuláció a valóság képzetét keltse, e két fő komponensnek összhangban kell működnie.

Olyan megoldást kerestem, amely viszonylag részletes terepet nyújt, ugyanakkor a hozzá kapcsolódó valósidejű számításokat egyszerűvé, a megjelenítést gyorssá teszi, ezáltal lehetővé téve, hogy a számítókapacitás nagy részét magára a szimulációra fordítsuk.

3.1 A terepmodell elkészítése

A terepet egyetlen felület alkotja, melynek kontrollpontjait az [x, z] síkra merőlegesen levetítve egy szabályos rácsot kapunk. A terepmodellezésben bevett szokás, hogy a magasságadatokat egy magasságtérképet tartalmazó fájlból töltik be. Ez a fájl sokszor egy szürkeskálás kép. A magasságtérkép tehát a kontrollpontok y komponenseit tartalmazó mátrix. Bár az így előállított terep túlságosan szögletesnek is tűnhet, ez orvosolható, ha a meglévő pontok segítségével valamilyen paraméteres (például NURBS-) felületet állítunk elő. A megjelenítéshez azonban ezt a felületet ismét poligonokra kell bontanunk. Felvetődik a kérdés, hogy hogyan lehetne elkerülni a közbülső lépést, azaz a paraméteres felület előállítását úgy, hogy mégis viszonylag sima felülethez jussunk. Erre a kérdésre ad választ a felosztott felületek módszere.

Ehhez nagyon hasonló módon töröttvonalat is simíthatunk. Ebben az esetben a vezérlőpontok közé felvesszük a szomszédos pontokat összekötő szakaszok felezőpontjait, majd a régi kontrollpontokat úgy módosítjuk, hogy az új helyük az eredeti helyük ½ súllyal, a két oldalukon lévő felezőpontok pedig ¼-¼ súllyal vett átlaga legyen.

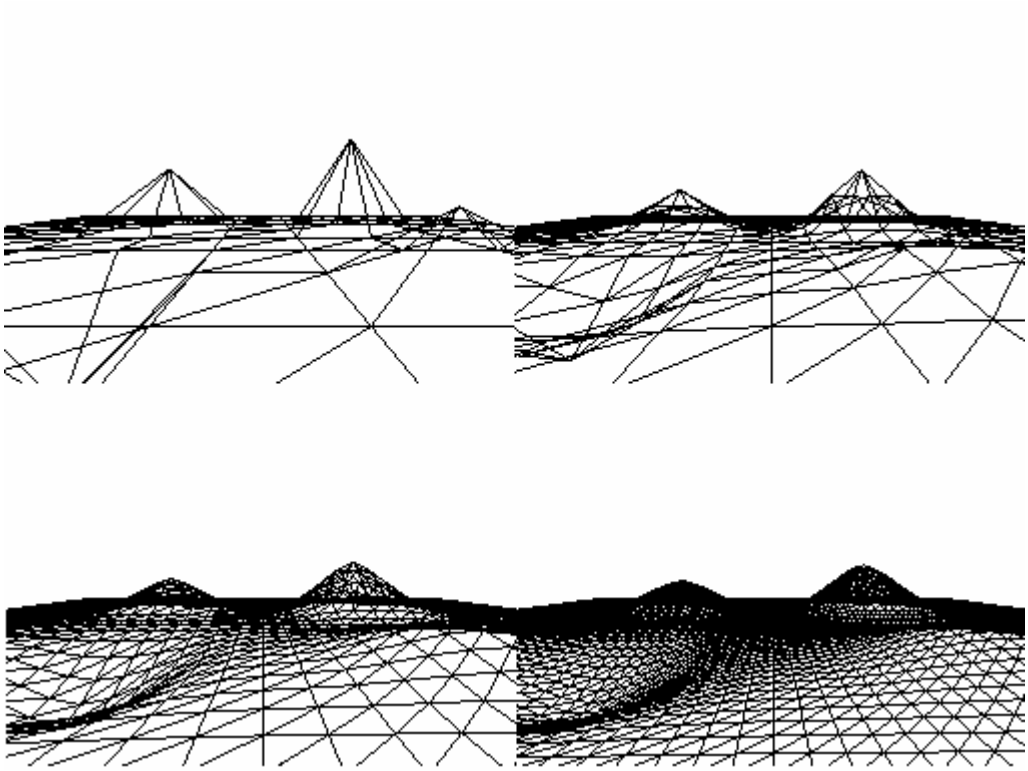


1. ábra: A módszer háromszori alkalmazása töröttvonal simítására

Látható, hogy ezt egyszer végrehajtva egy n pontból álló töröttvonalon, egy $2n-1$ pontból álló, simább töröttvonalat kapunk. Ha ezt végtelenszer ismételnénk, éppen B-spline-t kapnánk. Ezen eljárás háromdimenziós hálókra történő kiterjesztésének eredménye a Catmull-Clark felosztott felület.

Az algoritmus inputja a magasságtérkép, mely tehát egy n×n-es mátrix. Az egyszerűség kedvéért kvadratikus mátrixszal dolgozom, és az algoritmus jellege miatt az outputként kapott magasságtérkép is ilyen mátrix lesz. Ugyanis egy ekkora mátrixon egyszer végrehajtva az algoritmust egy $m \times m$-es mátrixot kapunk, ahol $m = 2n-1$. A szemléletesség kedvéért a kiindulási mátrixot tekintsük négyszöghálóként. Első lépésben vegyük fel minden él felezőpontját, továbbá minden négyszöglap közepén egy új pontot a négy csúcspont átlagaként. Nevezzünk ezeket a pontokat él- illetve lappontoknak. A második lépésben az élpontokat módosítjuk úgy, hogy az élpont az él két végén lévő csúcspont és az él két oldalán lévő két lappont átlaga legyen. Végül az eredeti csúcspontok új helyét súlyozott átlagként kapjuk meg, melyben az eredeti csúcspont ½ súllyal, a

kapcsolódó élek négy módosított élpontja és az illeszkedő lapok négy lappontja $\frac{1}{16}$ súllyal vesz részt. Az így kapott négyszöghálóra ez az eljárás ismét végrehajtható, és célszerű addig ismételni, amíg kellő simaságú felületet nem kapunk.



2. ábra: Ugyanazon kontrollpontok által leírt terep 0, 1, 2 és 3 simítás után

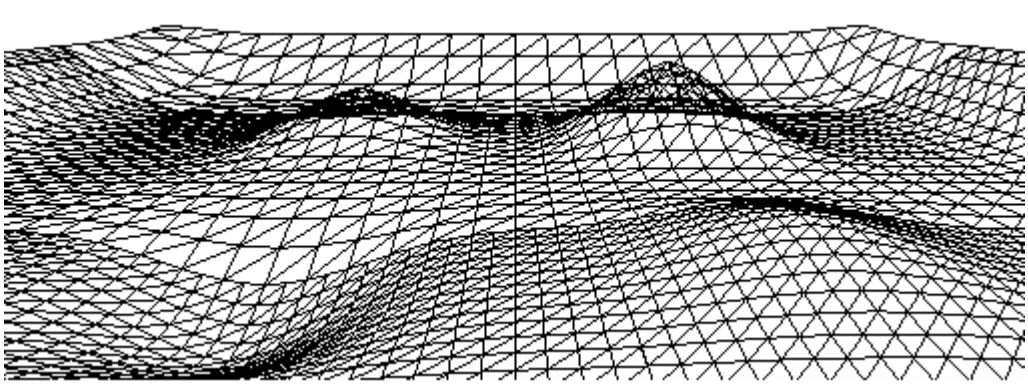
A terep előállításához a magasságtérképen kívül egyéb információkra is szükség van, ezeket a terepet leíró konfigurációs fájlból olvasom be. Ez a szöveges állomány tárolja a kontrollpontok magasságkomponenseit tartalmazó mátrixot, ennek méretét, a kontrollpontok távolságát, továbbá azt, hogy a fent ismertetett módszert hányszor ismételje a program a végleges terep elkészítéséhez.

3.2 A teljesítmény javítása

A terepet háromszögekre célszerű bontani, mert grafikai szempontból könnyebben kezelhetők, mint a négyszögek. A realisztikusabb hatás érdekében textúrát is szeretnénk a terephez rendelni. Látható továbbá, hogy a fent ismertetett módszer minden iterációja megnégyszerezi a terep poligonszámát, mely a sok kontrollpont vagy a több iterációban

történő simítás következtében igen nagyra nőhet. Mindezek következtében komoly teljesítménybeli problémák jelentkezhetnek.

Észrevehető azonban, hogy a nézőponttól távolodva a poligonok képernyőre eső vetületének területe egyre csökken. Így előfordulhat, hogy egy poligon képe kisebb, mint egy pixel, ezért a kamerától távol eső dolgokat fölösleges teljes részletességgel kirajzolni. Ezt az elvet használja ki a LOD (Level of Detail)-technika. Ennek lényege, hogy az objektumokról különböző részletességű modelleket tárolunk, és a kamerától mért távolság alapján választjuk ki, hogy melyiket jelenítsük meg.



3. ábra: A LOD-technika szemléltetése

A simítási eljárás után a végleges terep különböző részletességű modelljeit készítjük el. Például minden második pontot kihagyva lényegesen kevesebb, de nagyobb poligonból álló felületet kapunk. Mivel a terep a szimulációs ciklus elindítása után már nem változik, célszerű a simítás befejezése után minden szinthez elkészítenünk egy-egy rajzolási listát, mely már a textúra-leképezéseket is tartalmazza, és melyet a textúrával együtt a grafikus kártya tárol és jelenít meg, amikor erre parancsot adunk. Mivel a szimulációhoz szükséges számításokat mindig az eredeti, legrészletesebb terepmodellen kell elvégezni, az operatív memóriában csak az ehhez tartozó poligonhálót szükséges tárolni. Megjelenítéskor a virtuális teret a kamera pozíciójától meghatározott távolságra lévő, a nézési irányra merőleges síkokkal feldaraboljuk. A megfelelő térrészben a megfelelő részletességű terepmodellt jelenítjük meg, miközben a síkokkal történő elvágást is a grafikus egységre bízzuk. Ezáltal a CPU a szimulációs ciklus elindítása után mentesül a tereppel kapcsolatos műveletek nagyobb részének végrehajtása alól, értékes idejét így nagyrészt magára a szimulációra fordíthatjuk.

4 A jármű modellje

A hétköznapi életben sokféle szárazföldi járművel találkozhatunk. Munkám során, mivel ezek a legelterjedtebbek, négykerekű, elsőkerék-kormányzású autókkal foglalkoztam, melyeknek karosszériája merevnek tekinthető, tehát a kormányzásért kizárólag az első tengelyen lévő kerekek felelősek. Következésképp a jármű bal- és jobboldali kerekei nem azonos nyomon haladnak kanyarodáskor.

Az autó a szimulációban öt merev testtel reprezentálható, ezek az alváz és a négy kerék [9] [10]. Az alváznak hat szabadsági foka van, mely arra utal, hogy a mozgása leírható három tengely mentén történő elmozdulás és azok körül történő elfordulás kombinációjaként. A hátsó kerekeknek két szabadsági foka van: képesek a karosszérián függőlegesen elmozdulni és forgástengelyük körül elfordulni. Az első kerekek ezek mellett a kormányzás következtében a függőleges tengelyük körül is elfordulhatnak, így három szabadsági fokkal rendelkeznek. A teljes autó tehát összesen 16 szabadsági fokkal jellemezhető.

4. ábra: Az autót alkotó merev testek

Hogy kellően részletes és valósághű autót lássunk a szimulációban, modelljét célszerű valamelyik háromdimenziós modellezőprogram (például 3D Studio Max) segítségével elkészíteni és elmenteni, a szimulációs alkalmazáshoz pedig írni egy betöltőosztályt, mely az elmentett fájlt feldolgozva képes felépíteni a modellt, melyet ezek után már felhasználhatunk a virtuális világban is. Választásom a modellt tartalmazó fájl formátumára nézve a *3ds*-re esett, mert a legtöbb modellező program ismeri, viszonylag könnyen kezelhető, mégis realisztikus modellek leírására alkalmas.

Mivel a kerekek elmozdulhatnak a karosszérián és foroghatnak is, a karosszéria és a kerekek modelljét külön objektumként kell kezelni, és külön fájlban is kell megadni őket. Ugyanakkor elegendő egy kerék modelljét tárolni, majd a megfelelő transzformációk elvégzésével ezt négyszer kirajzolni, hogy az autó mindegyik kereke megjelenjen. Feltételeztem, hogy az alváz és a kerekek nem deformálódnak, vagyis modelljeik a terephez hasonlóan nem változnak a szimuláció során. Ezért modelljeik betöltése után ezekhez is felépíthető egy-egy rajzolási lista. Kirajzoláskor meg kell adni a szükséges transzformációkat, majd végrehajtatni a rajzolási listát. A kerékhez tartozó listát tehát minden kirajzolás során négyszer kell meghívni, a hívás előtt pedig a megfelelő kerékhez tartozó eltolásokat és elforgatásokat kell megadni. Miután a rajzolási listák elkészültek, az operatív tárban már nincs szükség a modellekre. Azonban a szimulációhoz szükséges jellemzőket, például a kerék sugarát vagy a tengelytávot megőrizzük a szimulációs számítások számára.

Az autó összes jellemzőjét a hozzá tartozó konfigurációs fájlból olvasom be. Ez a szöveges állomány tartalmazza a karosszéria és a jobb első kerék modelljét tartalmazó 3DS fájlok elérési útvonalát, valamint a jármű egyéb jellemzőit, például a kerekek helyzetét az alvázon, a kerekek sugarát, a felfüggesztési rendszer jellemzőit és a motor nyomatékát.

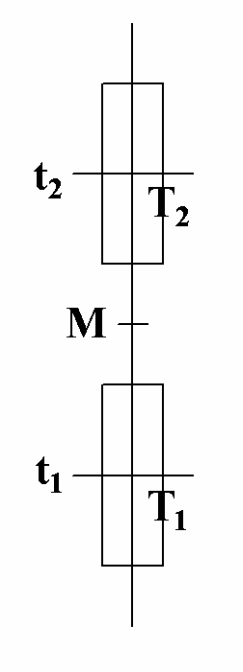
4.1 A jármű mozgásának modellezése

Bár a szimulációs alkalmazásban nem jelenik meg, egyszerűsége miatt tekintsük előbb a kétkerekű járművet. Hasonló megfontolásból a mozgás modellezésének ismertetése során kezdetben vonatkoztassunk el a motortól, a pedáloktól, a felfüggesztési rendszertől és a

járműre ható külső erőktől, a sebességet és a kormányállást tekintsük állandónak, a terepet pedig síknak.

4.1.1 Kétkerekű jármű mozgása sík felületen

Mielőtt a négykerekű jármű mozgásának ismertetésére térnénk, vegyünk tehát egy egyszerűbb konstrukciót, a két tengellyel rendelkező kétkerekű járművet. Az első tengelyen lévő legyen a kormányzott kerék.



Az ilyen járművekre jellemző oldalirányú dőléssel és annak hatásaival nem foglalkozom, mivel a vizsgálni kívánt négykerekű jármű nem képes ilyesmire.

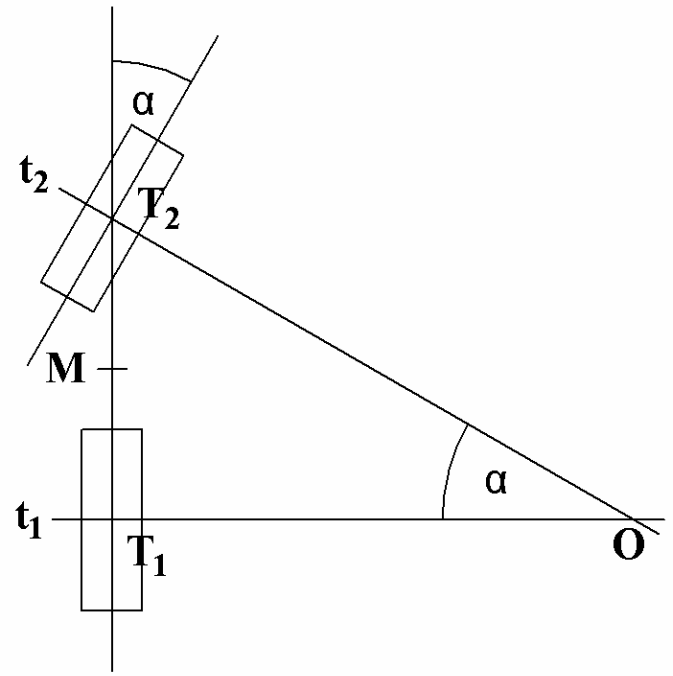
A hátsó tengely egyenesét jelölje t_1, az elsőét t_2, a tengelyek felezőpontjait T_1 illetve T_2. Ezek távolsága lesz a jármű tengelytávja, melyet jelöljön d. A jármű pozícióját adja meg a $\underline{h}$ vektor, mely a járműnek egy olyan kitűntetett pontjába mutat, melynek helyzete a járművön belül nem változik. Ez a pont legyen az egyébként is kiemelt szerepű tömegközéppont, melyet jelöljünk M-mel. Az egyszerűség kedvéért ez legyen éppen a T_1-et és a T_2-t összekötő szakasz felezőpontja. Ez azonban még nem határozza meg a jármű pillanatnyi állását, melyet például azzal az $\underline{l}$ vektorral adhatunk meg, mely a ($\underline{e}$-$\underline{h}$) vektorral párhuzamos egységvektor, ahol $\underline{e}$ a T_2 világ-koordinátarendszerbeli helyére mutató helyvektor. Ismernünk kell továbbá a jármű pillanatnyi sebességét (v) és első kerekének állását. Ha a jármű előrefelé halad, a sebessége pozitív, ha hátrafelé, akkor negatív. Az első kerék állását megadhatjuk azzal az α szöggel, mely megmutatja, hogy hány fokkal van az első kerék elfordítva. Tehát ha a kerék egyenesen áll, akkor az α szög 0, ha jobbra van fordítva, akkor $\alpha < 0$, míg ha balra, akkor $\alpha > 0$. Az α nem lehet akármekkora, a maximális befordítási szöget jelölje δ. Ekkor $-\delta \le \alpha \le \delta$.

A jármű aktuális állapotát a $(\underline{h}, \underline{l}, v, \alpha)$ négyessel adhatjuk meg. Feltételezhetjük, hogy a t-edik időpillanatban ismerjük az állapotváltozókat, és szeretnénk meghatározni az értéküket a $t' = t + \Delta t$ időpillanatban. Feltételezzük továbbá, hogy a Δt idő eltelte alatt a v és az α értéke állandó, ezért az új állapotot leíró négyest $(\underline{h}', \underline{l}', v, \alpha)$ alakban keressük.

Azt is feltehetjük, hogy $v \neq 0$, hiszen máskülönben a jármű nem mozdulna el. A jármű különböző pályán fog mozogni attól függően, hogy az első kerék egyenesen áll-e, ezért eszerint két esetet különböztetünk meg.

Az első esetben a jármű első kereke egyenesen áll, azaz $\alpha = 0$, tehát a tengelyek egyenesei párhuzamosak. Ekkor a jármű egyenes pályán fog mozogni, ezért az állása nem fog változni, azaz $\underline{l}' = \underline{l}$. Az új pozíciót pedig a következőképpen kaphatjuk meg: $\underline{h}' = \underline{h} + \Delta t \cdot v \cdot \underline{l}$.

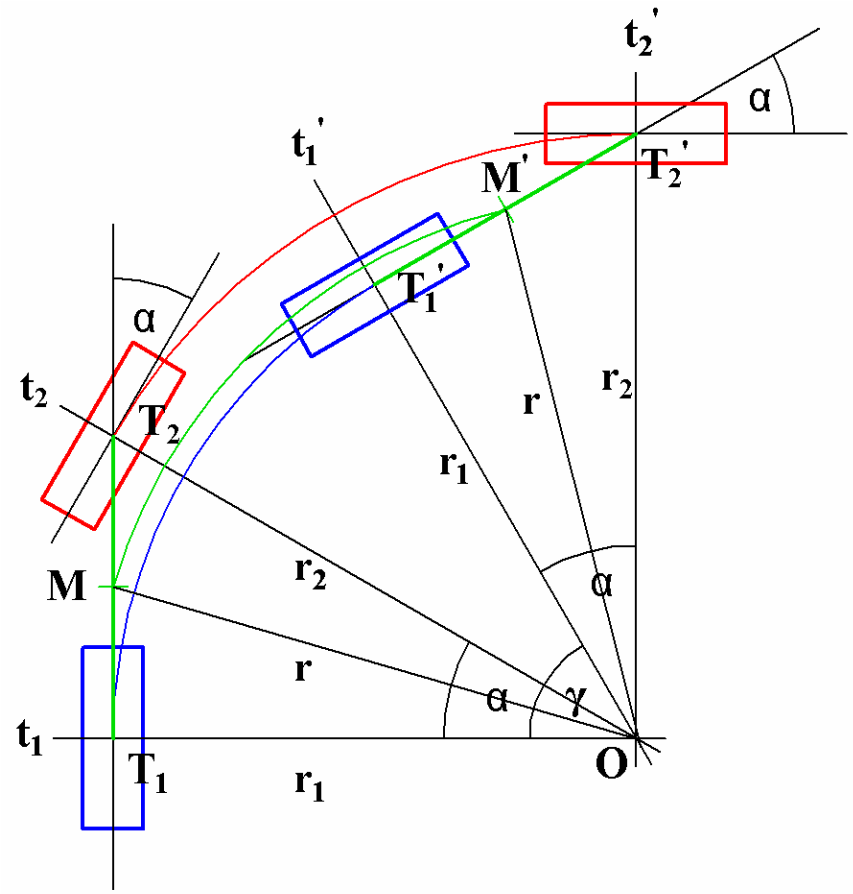
A másik esetben $\alpha \neq 0$. Ebben az esetben a tengelyek egyenesei metszik egymást, jelölje *O* a metszéspontot. Látható, hogy ezek az egyenesek éppen α szöget zárnak be.



5. ábra: A fordulási középpont kétkerekű jármű esetén

A jármű bizonyos idő eltelte után visszaérkezne ugyanebbe a helyzetbe, így látható, hogy Δt ideig *O* középpontú körpályán fog haladni. Azonban a T_1, T_2 és *M* pontok pályájának sugara különböző. A T_1 $r_1 = |OT_1| = \frac{d}{tg\alpha}$, a T_2 $r_2 = |OT_2| = \frac{d}{\sin\alpha}$, az *M* pedig $r = |OM| = \sqrt{\left(\frac{d}{2}\right)^2 + r_1^2}$ sugarú körpályán fog mozogni.

Hogy meghatározzuk, mekkora szöget fordul a jármű *O* körül, szükségünk lesz a szögsebességre: $\omega = \frac{v}{r}$. A γ elfordulási szöget pedig a $\gamma = \text{sgn}(\alpha) \cdot \omega \cdot \Delta t$ képletből kapjuk, a T_1, T_2 és *M* pont tehát ekkora szöget fordul el *O* körül. Hogy $\underline{l}'$ –t megkapjuk, az $\underline{l}$-et γ szöggel kell elforgatni az origó körül, $\underline{h}'$-t pedig úgy kaphatjuk meg, hogy $\underline{h}$-t eltoljuk -$\underline{o}$-val, majd elforgatjuk γ szöggel az origó körül, végül eltoljuk $\underline{o}$-val, ahol $\underline{o}$ az *O*-ba mutató helyvektor.



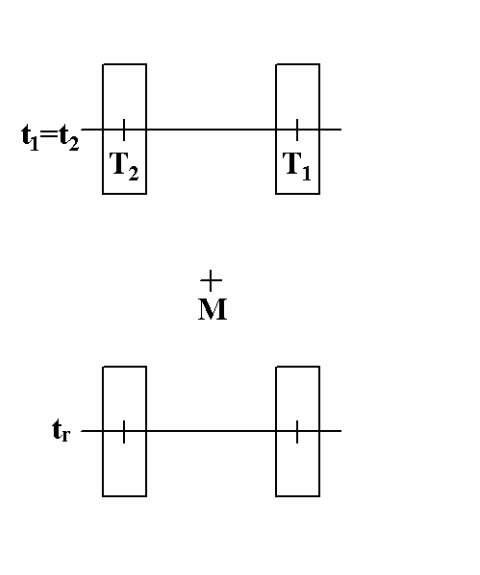
6. ábra: Kétkerekű jármű fordulása

A T_1 pont kisebb sugarú körpályán mozog, mint a T_2, ez okozza a többi járműnél is az úgynevezett farsöprés jelenségét.

4.1.2 Négykerekű jármű mozgása sík felületen

Olyan négykerekű járművet tekintünk tehát, amelynek első tengelyén lévő kerekei felelősek a kormányzásért. A hátsó tengely egyenesét jelölje t_r, a jármű

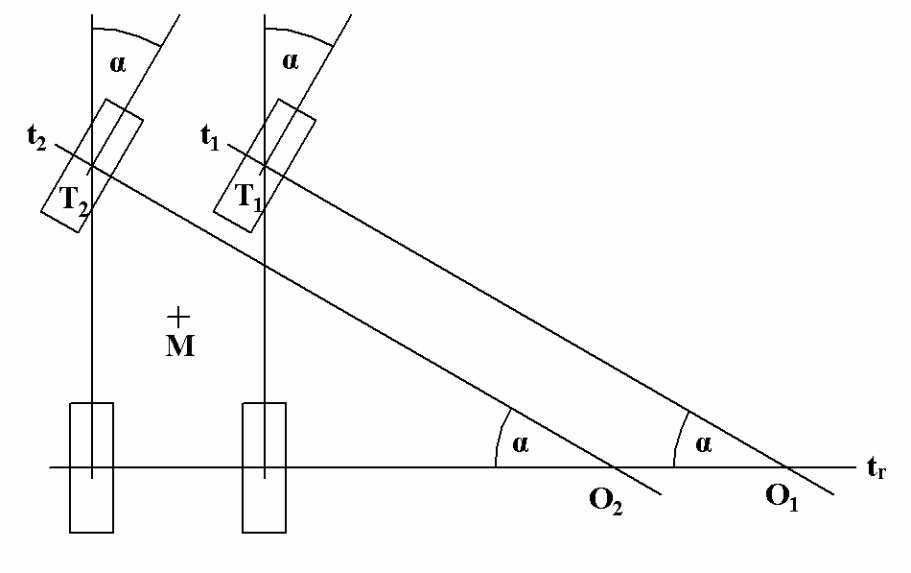
tömegközéppontját *M*, az első kerekek állását megadó szöget pedig α. Az első kerekek középpontja legyen T_1 és T_2, az első tengely egyenesének a T_1 illetve a T_2 körüli, α szöggel történő elforgatásával kapott egyeneseket jelöljük t_1-gyel illetve t_2-vel.



7. ábra: A vizsgált négykerekű jármű

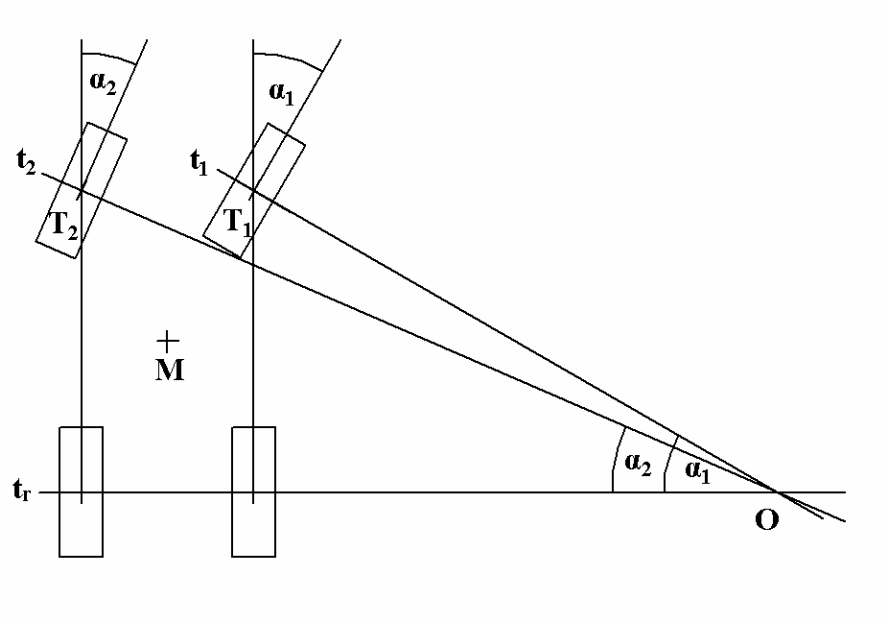
Ha $\alpha = 0$, azaz a kormány egyenesen áll, a jármű úgy fog viselkedni, mint az előbbi kétkerekű jármű hasonló esetben.

Ha viszont $\alpha \neq 0$, bonyolultabb helyzet áll elő. Jelölje O_1 a t_r és a t_1, O_2 a t_r és a t_2 egyenesek metszéspontját. Látható, hogy a jobboldali kerekek O_1, míg a baloldaliak O_2 körüli fordulásra törekednek, azonban a jármű nyilvánvalóan csak egy pont körül fordulhat el, melyet jelöljön *O*. Egy túlkormányzott jármű esetén ez a pont az ívbelső kerekek fordulási középpontja felé tolódik, míg egy alulkormányzott esetén az ívkülső kerekek fordulási középpontja felé.



8. ábra: A négykerekű jármű fordulási középpontjai

Éppen ezért az első kerekek általában nem ugyanolyan mértékben fordulnak el, hanem úgy, hogy O_1 és O_2 egybeessen. Így minden kerék azonos középpont körüli elfordulásra fog törekedni, ami miatt nem keletkezik erőhatás a karosszérián a két oldal széthúzása miatt.



9. ábra: Négykerekű jármű és fordulási középpontja

4.1.3 Négykerekű jármű mozgása egyenetlen felületen

Eddig a járművek mozgását csak sík felületen vizsgáltuk, a szimuláció viszont látványosabb, ha a jármű egyenetlen felületen halad. Az összképet azonban jelentősen ronthatja, ha a jármű kerekei szétcsúsznak, esetleg indokolatlanul elemelkednek a tereptől, vagy belesüllyednek abba.

A jármű mozgását ebben az esetben közelíthetjük oly módon, hogy az adott időpillanatban az előző alfejezetben ismertetett síkbeli mozgásnak tekintjük. A sík, amelyen az autó elmozdul, tekinthető a karosszéria aktuális vízszintes síkjának. A számítások egyszerűsítése végett célszerű nyilvántartani a karosszériához képest aktuálisan előre, jobbra és fölfelé mutató egységvektorokat, melyek a karosszéria relatív koordinátarendszerét alkotják. Az adott időpillanatban a mozgást tehát abban a síkban kell tekinteni, amelyet a karosszéria előre és jobbra mutató egységvektora határoz meg, és amely illeszkedik a jármű tömegközéppontjának pillanatnyi pozíciójára.

Amikor az autó $t' = t + \Delta t$ időpillanatbeli helyzetét akarjuk kiszámítani, feltételezhetjük, hogy a Δt érték igen kicsi, ezért tekinthetjük úgy, hogy ez idő alatt a jármű a t időpillanatbeli pozíciójában az előbb leírt módon meghatározott síkon mozog. Amennyiben a Δt érték a jármű sebességéhez viszonyítva nem túl nagy, azaz az autó a Δt idő alatt nem tesz meg túlságosan nagy távolságot, a számítás eredménye nem fog jelentősen eltérni a valóságostól.

Azzal azonban, hogy ezen a síkon kiszámoltuk a jármű új pozícióját, még nem a jármű tényleges *t'* pillanatbeli helyzetét kaptuk meg. Előfordulhat, hogy a kerekek nem illeszkednek a terepre, és a karosszéria új állását is meg kell határozni. Ezért a második lépésben a síkon kiszámolt pozícióban meghatározzuk a kerekek felfekvési pontjait a karosszériához viszonyított helyzetük és a karosszéria állása alapján. A következő lépésben a karosszéria helyzete alapján kiszámoljuk, hogy a jármű súlya hogyan oszlik meg a kerekek között, amiből a felfüggesztési rendszert ismerve meghatározható a rugók hossza (lásd a következő alfejezetben). Innen pedig megkapható, hogy elvileg hol vannak a karosszérián a kerekek felfüggesztési pontjai. Ha ezek megegyeznek az első lépés eredményéül kapott értékekkel, akkor megkaptuk a jármű új helyzetét. Ellenkező esetben a már kiszámolt adatok alapján meghatározzuk az új tömegközéppontot és a karosszéria síkját, majd a második lépésre ugrunk.

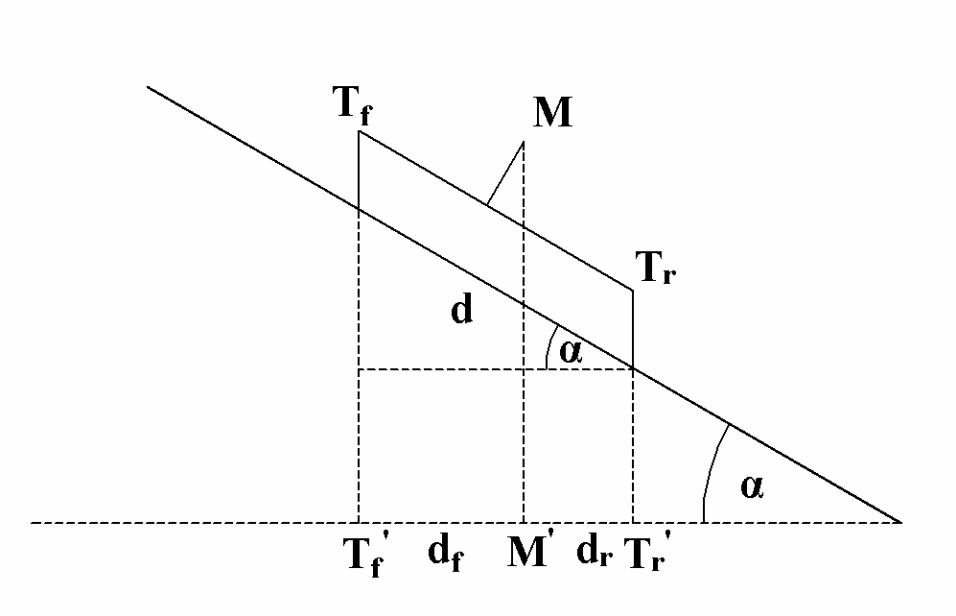
Az eljárást egy alkalmasan megválasztott hibahatárig célszerű folytatni, vagy az egyenletesebb futás érdekében előírt számszor teljesíteni a visszaugrást. A tapasztalat azt mutatja, hogy kellő simaságú terep mellett az új helyzetben elegendő egyszer kiszámolni a tömegközéppont új helyét és a karosszéria állását, mivel már így is megfelelő pontosság és ezzel realisztikus hatás érhető el, míg egy további visszalépés nem okoz észrevehető javulást.

Amennyiben változik a kormány vagy valamelyik pedál állása, azt a neki megfelelő állapotváltozó *t* időpillanatbeli módosításával jelezzük, és ennek megfelelően számítjuk ki a jármű $t' = t + \Delta t$ pillanatbeli jellemzőit. Ez szintén nem okoz érezhető eltérést a valóságos pályától, amennyiben a Δt értéke elegendően kicsi.

4.2 A felfüggesztési rendszer

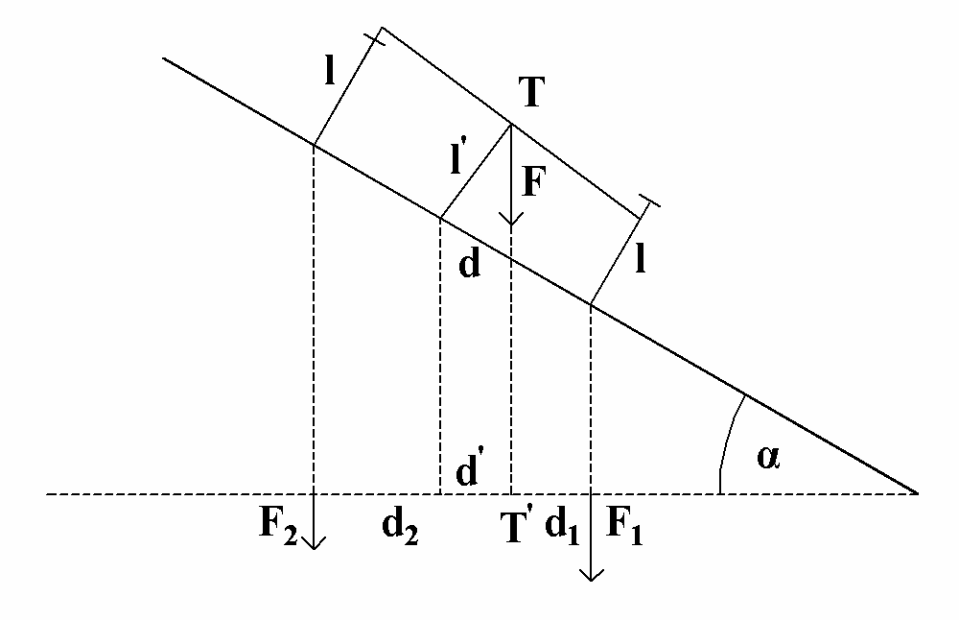
Hogy meghatározhassuk, hogyan oszlik meg a jármű súlya az egyes kerekek felfüggesztései között, ismernünk kell a karosszéria állását, a rugóhosszokat és a rugóállandókat. Viszonylag valósághű eredmény kapható azzal a kétlépéses számítási módszerrel, melynek első lépésében a két tengely között osztjuk el a jármű súlyát, majd a második lépésben az egyes tengelyeken lévő kerekek között osztjuk el az adott tengelyre előzőleg kiszámolt súlyt.

A kerekek felfekvési pontjai alapján kiszámítható a tengelyek helyzete. Mivel a tengelyek különböző szögben is megdőlhetnek oldalra, a súly első lépésbeli, tengelyek közti elosztását a jármű középvonalának állása alapján végeztem. Az első tengely felezési pontját jelölje T_f, a hátsóét T_r, a tömegközéppontot *M*, a tengelytávot *d*. A jármű középvonala és annak függőleges vetülete által bezárt szög legyen α, T_f függőleges vetülete T_f', T_r-é T_r', *M*-é M'. Az első és hátsó tengelyre ható F_f és F_r erők aránya $d_r : d_f$ lesz, ahol $d_f = \left|T_f' M'\right|$, $d_r = \left|M' T_r'\right|$.



10. ábra: A súly elosztása a tengelyek között

A második lépésben a fent kiszámított F_f és F_r súlyokat osztjuk el az első illetve a hátsó kerekek között úgy, ahogy ezt egy kétkerekű, egytengelyű jármű esetén tennénk. Legyen *d* a tengely hossza, d' a tengely merőleges vetülete, α ez utóbbi kettő által bezárt szög. A rugók hosszát alapterhelésnél (azaz amikor a jármű vízszintes sík felületen áll) jelölje *l*, a rugóállandót k_r, a tengelyre ható teljes erőt *F*, mely tehát az előbb kiszámolt F_f vagy F_r. Legyen $l' = l + (1 - \cos\alpha) \cdot F \cdot 2k_r$ a tengely középpontjának (*T*) magassága. A T' pont a tengely d' hosszúságú vetületét egy $d_1 = \frac{d'}{2} - l' \cdot \sin\alpha$ és egy $d_2 = \frac{d'}{2} + l' \cdot \sin\alpha$ hosszúságú részre osztja. Az $F_1 + F_2 = F$ és $d_1 F_1 = d_2 F_2$ egyenletek miatt $F_2 = \frac{d_1 F}{d'}$, $F_1 = F - F_2$. A rugók irányába ható megfelelő erőket pedig az $F_1' = F_1 \cdot \cos\alpha$ illetve $F_2' = F_2 \cdot \cos\alpha$ képletek alapján kaphatjuk meg. Innen a rugók megnyúlásának változása $\Delta l_1 = (F_1' - \frac{F}{4}) \cdot k_r$ illetve $\Delta l_2 = (F_2' - \frac{F}{4}) \cdot k_r$.



11. ábra: A súlyeloszlás számítása a jármű egyik tengelyén

Ezt a számítást mindkét tengelyre elvégezve a mind a négy rugóhossz megkapható. Ezek alapján kiszámítható a tömegközéppont új helyzete és a karosszéria új állása. Látható, hogy a kerekek négy felfüggesztési pontja nem mindig illeszkedik egy síkra, ezért a karosszéria új síkjának azt a síkot választottam, amely párhuzamos az egymással átellenesen elhelyezkedő kerekek felfüggesztési pontjaira illeszkedő egyenesekkel, és egyenlő távolságra van azoktól. A tömegközéppont új helyzete ezen sík és a karosszérián belüli relatív helyzete alapján meghatározható.

4.3 Kormányzás

A kormánnyal tehát az első tengelyen lévő kerekek állását szabályozhatjuk. Ahogyan a kétkerekű jármű esetében, itt is meg kell állapítani egy maximális szögértéket, ameddig az első kerekék fordítható. Hogy az első kerekek ilyen mértékű elfordulását elérjük, a kormányt ütközésig kell forgatni a megfelelő irányba.

Az autóknál a kormánykerék és a kormányzott kerekek elfordulási szöge általában nem egyezik meg. Ezek arányát a kormányzási arány adja meg, melynek értéke a közúti autók esetében legtöbbször 12:1 és 20:1 közé esik. Versenyautók esetében ez az arány lényegesen kisebb, míg a kétkerekű járművek, így a kerékpárok és a motorkerékpárok esetén mindig 1:1. Ha egy autó kormányzási aránya 12:1, az azt jelenti, hogy a

kormánykerék 360˚-os elfordítása a kormányzott kerekek 30˚-os elfordulását eredményezi.

Ismerni kell továbbá azt is, hogy adott kormányállásnál a két kormányzott kerék közül melyik mekkora szerepet játszik a fordulási középpont meghatározásában. Ebből a szempontból fontos szerepet játszik a jármű tengely- és nyomtávja is, amiket a kerekek tömegközépponthoz képesti helyzetéből is származtathatunk.

Fontosnak tartottam továbbá bevezetni azt a konstanst, amely megmutatja, hogy egységnyi idő alatt hány fokkal fordítható el a kormány. Ez az érték talán inkább jellemző az autó vezetőjére vagy robotjármű esetén a kormányt kezelő mechanizmusra, mint magára a kormányműre, logikailag mégis ide tartozik.

A fent említett, a számításokhoz nélkülözhetetlen adatokat a jármű konfigurációs fájlja tartalmazza.

4.4 Gyorsítás és lassítás

Az autó fék- és gázpedáljának állását egy-egy 0 és 1 közé eső valós számmal reprezentálhatjuk. A 0 érték jelenti azt, hogy a pedál teljesen fel van engedve, az 1 pedig azt, hogy teljesen be van nyomva. A konfigurációs fájl tartalmaz továbbá egy konstanst, mely megmutatja, hogy mennyi időt vesz igénybe, amíg a felengedett pedált teljesen benyomjuk. A motor nyomatékát konstansnak tekintettem, de a valóságban nyomatékgörbét szokás megadni.

Gyorsításkor a gyorsítóerőt az $F_{gy} = t \cdot \varepsilon \cdot F - v^2 \cdot \rho$ képletből kaphatjuk meg, ahol *t* a gázpedál állását reprezentáló 0 és 1 közötti valós szám, *F* a nyomatékkal arányos maximális gyorsítóerő, ε pedig az ehhez tartozó konstans. Gyorsításkor a motor gyorsítóerejének irányával ellentétes, a jármű sebességének négyzetével arányos lassítóerő lép fel, ezt fejezi ki a $v^2 \cdot \rho$ tag. A ρ a jármű légellenállási együtthatójával – melyet c_v-vel szoktak jelölni - arányos konstans. Utcai autók esetén a kisebb légellenállási együttható az előnyös, mert ezzel kisebb fogyasztás és jobb gyorsulás érhető el. Versenyautók esetén ez az érték sokkal nagyobb lehet, mert ezáltal nagyobb leszorítóerő keletkezik.

Fékezéskor a fékezőerő $F_f = -b \cdot \delta \cdot F - v^2 \cdot \rho$ alakban kapható, ahol b a fékpedál állását reprezentáló valós szám, F a maximális fékezőerő, δ pedig az ehhez tartozó konstans. Ebben az esetben a légellenállásból származó lassítóerő ugyanabba az irányba hat, mint az autó fékrendszere által keltett lassítóerő, ezért az összegben azonos előjellel szerepelnek.

Ha éppen egyik pedál sincs benyomva, akkor a járművet a motor fékezőereje és a légellenállás lassítja: $F_f = -\lambda \cdot F - v^2 \cdot \rho$.

A jármű sebességére hatással lehet azonban a terep is: ha lejtőn halad, gyorsítóerő, ha emelkedőn, fékezőerő lép fel. Az így fellépő erő $F_l = \sin\alpha \cdot M \cdot \varphi$ alakban számítható, ahol α a lejtő vagy emelkedő dőlésszöge, M a jármű tömege, φ pedig a jármű lejtőn és emelkedőn tapasztalható sebességváltozásának mértékét befolyásoló konstans.

A fenti képlet alapján kapott erőt a jármű előbb kiszámolt gyorsító- vagy fékezőerejéhez hozzáadva megkapható az F_e tényleges gyorsító- vagy fékezőerő. Az $a = \frac{F_e}{M}$ képlet alapján megkapható a pillanatnyi gyorsulás. Az előjelek helyességét szem előtt tartva a fenti számításokban a gyorsítóerők pozitív, a fékezőerők negatív előjellel szerepeltek, ezért a kapott gyorsulás előjele is helyes lesz. Ezek után a $\Delta v = a \cdot \Delta t$ képlettel számolható a sebességváltozás, melynek előjele az előbbiek miatt szintén helyes lesz.

5 A jármű vezérlésének automatizálása

Az előző fejezetben ismertetett járművet szeretnénk képessé tenni arra is, hogy az irányítását érintő felhasználói beavatkozás nélkül képes legyen egy előre megadott pozícióba eljutni. Ez úgy fog történni, hogy még mielőtt elindulna, a jármű kiszámít egy előzetes pályatervet, majd ehhez igazodva igyekszik elérni a célt. A pályatervet pályaszegmensek alkotják, melyeket ellenőrzőpontok választanak el egymástól.

Mivel a jármű szenzorai csak bizonyos távolságig képesek érzékelni a környezetet, a pályaterv elkészítését végző egység nem ismerheti teljesen a terepviszonyokat. Ezért az autó a pályatervet sík felületen fogja meghatározni. Az ellenőrzőpontok síkon kiszámolt helyzete azonban egyértelműen meghatározza a valódi terepen elfoglalt pozíciójukat: hogy ezt megkapjuk, mindössze egy függőleges eltolást kell alkalmazni. Ezért ez nem jelent túl nagy problémát, azonban mivel a jármű nem a számítás alapjául vett sík felületen, hanem a valódi terepen fog haladni, a terep egyenetlenségei és egyéb hatások miatt valamilyen mértékben nyilvánvalóan le fog térni a kiszámolt pályáról. Hogy a célt mégis elérje, korrekciót fog alkalmazni.

5.1 Az előzetes pályaterv elkészítése

A számítást tehát síkon végezzük, ami valamelyest egyszerűsíti a számításokat. Feltételezhetjük, hogy ismerjük a jármű kezdeti pozícióját és állását, valamint az elérni kívánt pozíciót és az ottani állást, illetve a jármű mozgását befolyásoló paramétereket: a tengely- és nyomtávot, a kormányzott kerekek maximális elfordulási szögét és elfordítási sebességét.

Feltételeztem, hogy a jármű az elindulása előtt tetszőleges kormányállást választhat, illetve azt is, hogy a célhelyzetben nincs meghatározva, hogy milyen kormányállással kell megérkeznie. A terv elkészítése alatt figyelmen kívül hagytam a járműre ható erőket, mert az autó nem a síkon, hanem a valós terepen fog haladni, ami azt eredményezi, hogy a terep egyenetlenségei miatt a jármű úgyis le fog térni a kiszámolt pályáról. Ezért elegendő egy némileg durvább terv elkészítése is, hiszen a járművet mindenképp fel kell készíteni arra, hogy amint érzékeli, hogy letért a kijelölt pályáról, megfelelően reagáljon a

helyzetre. Kevésbé pontos terv elkészítése egyszerűbb és gyorsabb, ugyanakkor nem lesz annyira pontatlan, hogy a jármű jelentősen letérjen róla, ezáltal veszélyeztetve akár azt is, hogy elérje a célt.

Könnyen belátható, hogy a járművet egy viszonylag közeli célpozícióba eljuttatni teljesen más szemléletmódot igényel, mint egy viszonylag távoliba. Ha figyelmen kívül hagyjuk, hogy a jármű tolatni is képes, akkor ez utóbbi esetben a pályaterv két kormánymozdulatot ír le: a jármű egy meghatározott kormányállással indul el előre, melyet bizonyos ideig tart, majd kiegyenesítni a kormányt, és egyenesen halad valameddig, ez követően bizonyos fokig befordítja a kormányt, végül ezt a kormányállást tartva halad, amíg el nem éri a célt. A paramétereket alkalmasan megválasztva a pályaterv a jármű lehető legrövidebb útját is leírhatja a célig. Ha a tolatást is megengednénk, bizonyos esetekben még rövidebb úton érhetnénk el a célt, de ennek az lehet az ára, hogy nagy távolságot kell tolatva megtenni. Ebből a szempontból helyénvalóbb mellőzni a tolatást. A másik esetben, amikor a célhelyzet viszonylag közel esik a kiindulási pozícióhoz, szintén megadható olyan pálya, ahol a járműnek két kormánymozdulatot kell tennie, és végig előrefelé kell haladnia, azonban ez a megtett utat tekintve korántsem optimális. Ebből a szempontból előnyösebb pályát kaphatunk, ha kihasználjuk, hogy a jármű tolathat is.

Tekintsük az egyszerűbb esetet, azaz amikor a járműnek egy viszonylag távoli helyre kell eljutnia tolatás nélkül. Az előbb leírt két kormánymozdulat alapján a pályát öt főbb szegmensre oszthatjuk: az első és utolsó szegmens konstans kormányállással való fordulást, a harmadik egyenes mentén történő mozgást ír elő, míg a második és a negyedik a kormány kiegyenesítését illetve befordítását foglalja magába. A kérdés csak az, hogy hol legyenek a szegmensek határai, azaz tulajdonképpen az, hogy milyen kormányállással induljunk és érkezzünk meg, illetve meddig tartsuk azt. A második és a negyedik szegmens ugyanis ezek után már egyértelmű, ha a terv elkészítése alatt az autó sebességét konstansnak tekintjük. Látható, hogy konstans kormányállás mellett a leírt pálya független a sebességtől (feltéve, hogy eltekintünk azoktól a hatásoktól, amelyek eltéríthetik a járművet), de ha forgatjuk a kormányt, a leírt pálya alakjára és hosszára hatással van a sebesség. Ezért a kiszámított pálya függeni fog a feltételezett sebességtől is, melyet a számítás során vehetünk végig konstansnak, azaz eltekinthetünk a gyorsítástól és a lassítástól. Ezt megtehetjük, ha a pályaszámításkor vett sebességnél a szimuláció

elindítása után a jármű nem halad gyorsabban, miközben a kormányt mozgatja. Ez azért fontos, mert adott sebesség mellett számított íven kisebb sebességgel haladó járművet könnyű a pályán tartani, ha viszont a jármű gyorsabban halad, várhatóan le fog térni a kijelölt útvonalról. Például, ha a második pályaszegmenst tekintjük, a kormány mindkét esetben ugyanannyi idő alatt egyenesíthető ki, azonban nagyobb sebesség mellett nagyobb lesz az ezen idő alatt megtett út is. Ha a sebességünk kisebb a pályaszámítás alatt alkalmazottnál, egyszerűen csak lassabban kell forgatni a kormányt, ha viszont nagyobb, gyorsabb kormánymozdulatokra lenne szükség, amit a fizikai korlátok lehetetlenné tesznek, emiatt eltérésre kell számítanunk a pályatervtől. Elképzelhető persze, hogy ezt az eltérést a terep egyenetlenségei vagy egyéb hatások kompenzálják majd, így a jármű haladhatna akár nagyobb sebességgel is, de ebben nem lehetünk biztosak. Ezért célszerű a szimuláció közbeni javasolt utazósebességet használni a pályaterv elkészítése során. Ezt a sebességet jelölje *v*. A számításokhoz szükség lesz továbbá egy Δt értékre, amely a *v*-vel együttesen meghatározza a pályaterv pontosságát.

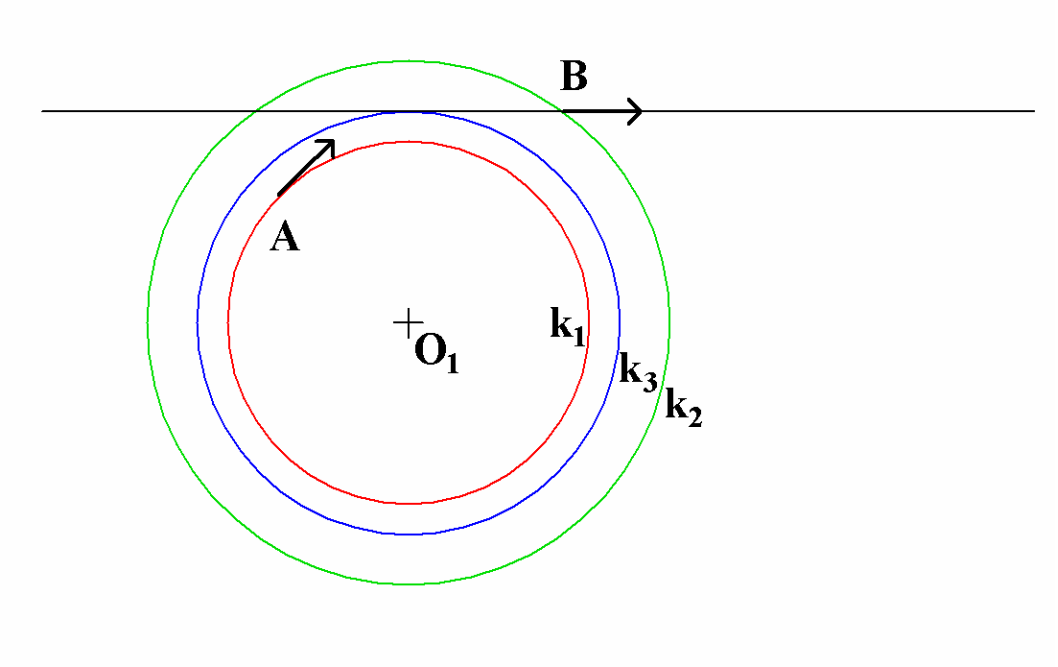
A jármű kezdeti pozíciója legyen H_1, állását adja meg az l_1 vektor, a célpozíció legyen H_2, az ottani állás pedig az l_2 vektorral adott. A jármű kormányzott kerekeinek állását a kezdeti helyzetben jelölje az α_1, a célhelyzetben az α_2 szög.

A pályát a következőképpen kaphatjuk meg. Kezdetben α_1 és α_2 legyen 0, azaz a jármű egyenes pályán indul el, és egyenes pályán érkezik is meg. Ha ez a két egyenes egybeesik, megtaláltuk a pályát, mely tehát egyetlen egyenes szakaszból áll. Ellenkező esetben meghatározzuk a végleges α_1 és α_2 előjelét: ha a H_1 pont és az l_1 irányvektor által megadott egyenes jobb oldalán helyezkedik el a H_2 pont, akkor előbb jobbra fogunk fordulni, azaz α_1 előjele negatív, különben előbb balra fordulunk, azaz α_1 előjele pozitív. Hasonló módon a H_1 pontnak a H_2 pont és az l_2 irányvektor által meghatározott egyeneshez képesti helyzetéből meghatározzuk α_2 előjelét. Majd α_1-et és α_2-t az előjelüktől függően csökkentjük vagy növeljük egy alkalmas φ értékkel. Ezután kiszámítjuk, hogy melyik az az egyenes, amelyen a jármű tovább haladna, ha a kiindulási helyzetből indulva kiegyenesítené az α_1 szögben álló első kerekeket. Hasonlóan kiszámítjuk azt az egyenest is, amelyen a járműnek haladnia kellene, hogy a célhelyzetbe megérkezzen α_2 szögig befordítva a kormányt. Ha ez a két egyenes egybeesik, megtaláltuk a pályát. Különben tovább kell folytatni α_1 és α_2 növelését vagy

csökkentését. Azonban a kerekek nem fordíthatók akármeddig: $-\delta \le \alpha_1, \alpha_2 \le \delta$ kell legyen, ahol δ az első kerekek maximális befordítási szöge. Ezért ha elérjük a δ illetve $-\delta$ értéket, nem növelhetjük illetve csökkenthetjük tovább α_1 és α_2 értékét. Ekkor azt a szöget kell növelni, amit a jármű az elindulása után a fordulási középpont körül fordul teljesen befordított, konstans kormányállással, illetve amit a jármű a célba érkezése előtt ugyanilyen módon fordul még az ottani fordulási középpontja körül. Ezeket a szögeket jelölje β_1 és β_2.

Ennek a megoldásnak van egy nagy hibája: előfordulhatnak olyan esetek, amikor az indulás és az érkezés oldalán nem mindig egyszerre kellene változtatni a szögeket, ami nagyban megnehezíti a helyzetet. Emellett a lehetséges megoldást valamekkora hibával kell elfogadni, mert a két egyenes általában nem fog pontosan egybeesni. Ha azonban elgondolkodunk ezen algoritmus fejlesztési lehetőségein, és elsősorban arra koncentrálunk, hogy mikor melyik oldalon lévő α és β szögeket kellene változtatni, kidolgozható egy gyorsabb és geometriai megközelítése miatt elegánsabb megoldás.

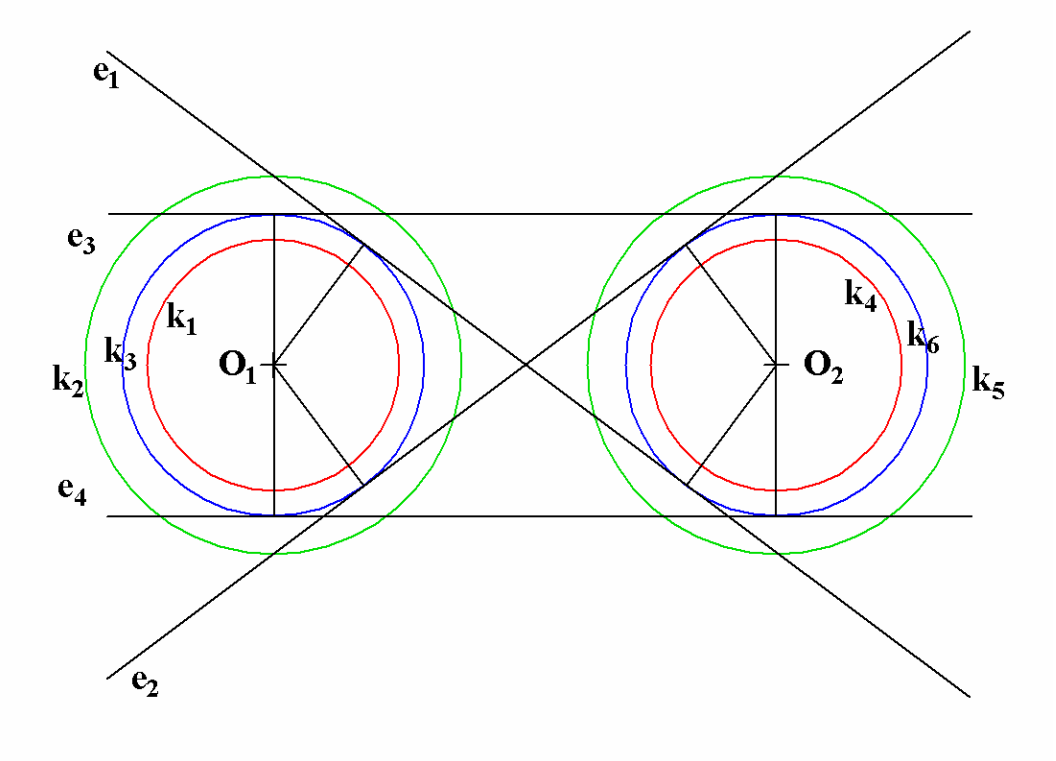
Ha a pálya egyetlen egyenes szakasszal megadható, az az eset ugyanúgy külön kezelendő. Ellenkező esetben az α_1 és α_2 szögek előjelét a fentebb leírt módon határozzuk meg ebben a megoldásban is. Tekintsük a járművet a kiindulási helyzetben az erre az oldalra kiszámított irányba maximálisan elfordított kormányállással, melyet tartva az autó körpályán fog mozogni, aminek sugara és középpontja egyszerűen meghatározható (lásd 4.1.2 fejezet). Ezen körpályát jelöljük k_1-gyel, a középpontja legyen O_1, A pedig egy tetszőleges pontja. A jármű sebességét a megadott v konstansnak tekintve meghatározhatjuk a B pontot, ahová a jármű eljut, amíg az A pontból indulva kiegyenesíti a kormányt. Látható, hogy a fordulási kör minden egyes pontjához egyértelműen tartozik egy-egy ilyen pont, melyek szintén egy O_1 középpontú kört alkotnak, jelöljük ezt k_2-vel. A jármű a B pontba érkezve egyenesen haladhat tovább. Az ezen egyenest érintő, O_1 középpontú kört jelölje k_3. Mindez tehát azt jelenti, hogy ha a k_1 körpályán v sebességgel haladó autó kiegyenesíti a kormányt, akkor olyan egyenes pályára fog állni, mely érinti a k_3 kört.



12. ábra: A három kör a kiindulási oldalon

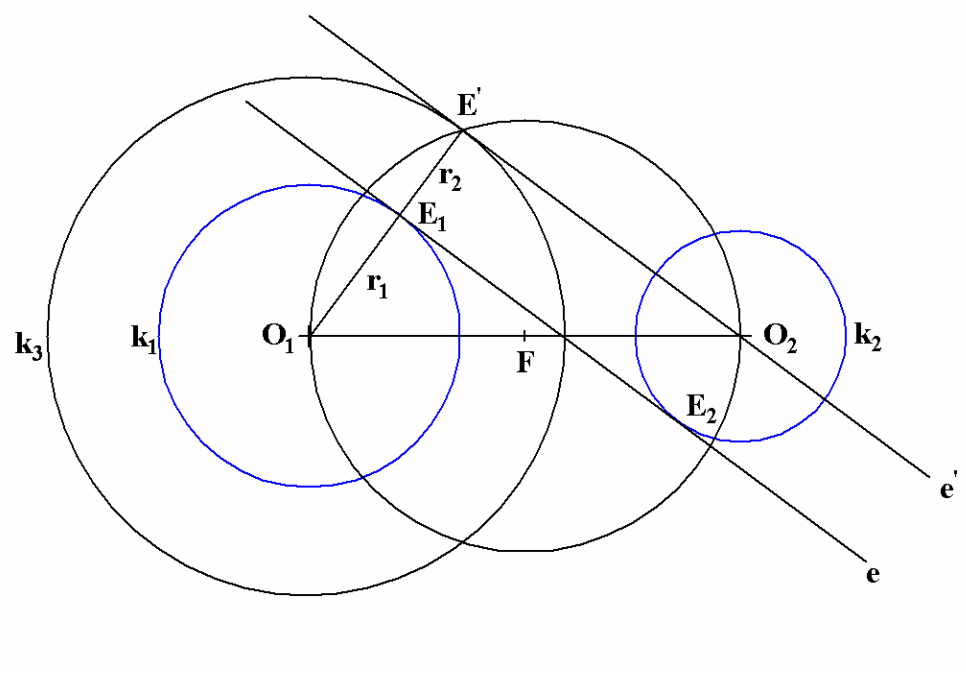
Hasonló módon a célhelyzetnél is meghatározható három ilyen kör, itt azonban a jármű előrefelé haladó mozgását visszafelé kell lejátszani, hiszen innen nem elindul az autó, hanem ide érkezik meg. A fordulási kör ugyanaz lesz itt is, ezt jelölje k_4, középpontját O_2. A k_5 kört alkossák azok a pontok, melyekből, ha a járművet egyenes kormányállásból, a kormányt folyamatosan befordítva elindítjuk, az autó a k_4 kör egy pontjára érkezik meg, mire a kormányt teljesen befordítja, és ezen a körön is halad tovább, ha engedjük. Látható, hogy k_5 középpontja is O_2 lesz. Jelölje k_6 azt a kört, amelynek középpontja O_2, és érintője az előbbi egyenes, amelyen a jármű haladt, amíg el nem kezdte befordítani a kormányt. Ahhoz tehát, hogy a jármű a k_4 fordulási körre megérkezzen, a k_6 kör egy érintője mentén kell egyenesen haladnia, majd egy alkalmas pontban elkezdenie a kormány befordítását.

Minderre azért van szükség, mert a pályaterv harmadik szegmensét a k_3 és a k_6 körök valamelyik közös *e* érintőjének egyenesére fogjuk illeszteni. A két körhöz két-két belső és külső érintő szerkeszthető, ezek közül azonban csak egy lesz a célnak megfelelő. α_1 és α_2 már meghatározott előjelei alapján eldönthető, hogy külső vagy belső érintőt kell-e keresnünk: ha a két előjel megegyezik, külső, ellenkező esetben belső érintő lesz alkalmas. A megmaradt két érintő közül szintén a fordulási irányok alapján választhatjuk ki a nekünk megfelelőt.



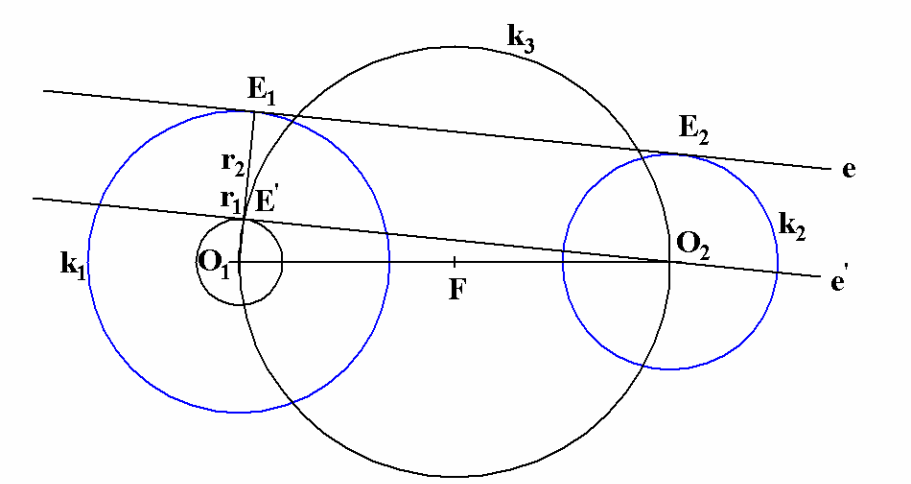
13. ábra: A k_3 és k_6 körök közös érintői

Az kérdéses érintő viszonylag egyszerűen meghatározható. Mind belső, mind külső érintő esetén a feladat visszavezethető körhöz külső pontból húzott érintő megszerkesztésére. Ismerjük a körök középpontjait (O_1 és O_2) és sugarait (r_1 és r_2), továbbá tudjuk, hogy melyik érintő lesz számunkra megfelelő. Ha belső érintőre van szükségünk, előbb az O_2 pontból az O_1 középpontú, $r_1 + r_2$ sugarú körhöz húzott e' érintőt határozzuk meg. Az E' érintési pont ennek a nagyobb körnek és az O_1O_2 szakasz fölé írt Thalész-körnek a megfelelő metszéspontja lesz. Az e érintő pedig az O_1-től r_1, e'-től r_2 távolságra lévő, e'-vel párhuzamos egyenes lesz.



14. ábra: Belső érintő szerkesztése

Külső érintő esetén ehhez hasonlóan kell eljárni. Tegyük fel, hogy $r_1 > r_2$. Előbb az O_2 pontból az O_1 középpontú, $r_1 - r_2$ sugarú körhöz húzott e' érintőt határozzuk meg. Innen megkapható az eredeti köröket érintő, e'-vel párhuzamos e érintő. Amennyiben $r_1 < r_2$, egyszerűen csak meg kell cserélni a köröket. Előfordulhat olyan eset, amikor a körök sugara megegyezik. Ez az eset külön kezelendő ugyan, de egyszerűbb, mivel a kérdéses érintő párhuzamos lesz a körök középpontjaira illeszkedő egyenessel, és $r_1 = r_2$ távolságra lesz attól.

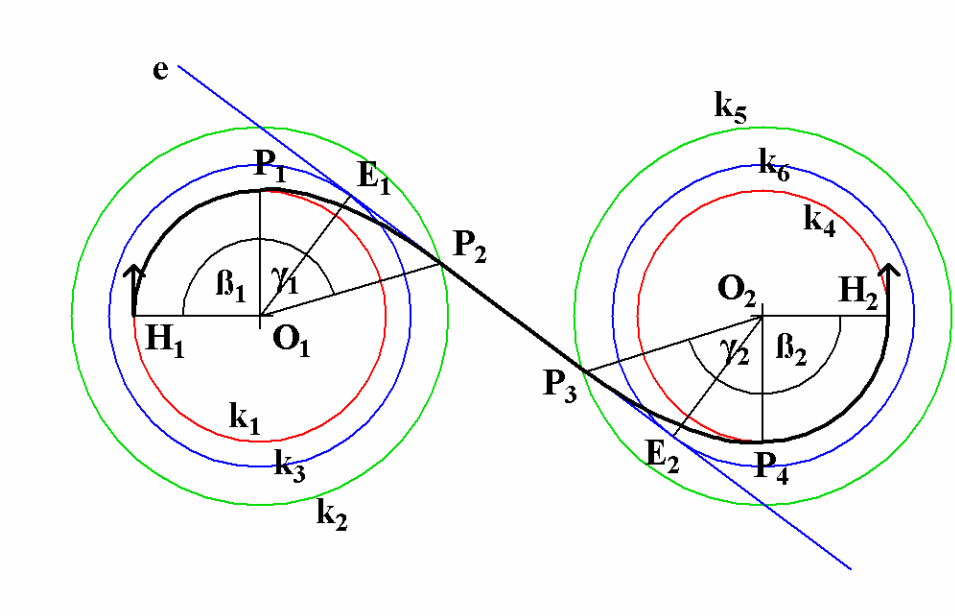


15. ábra: Külső érintő szerkesztése

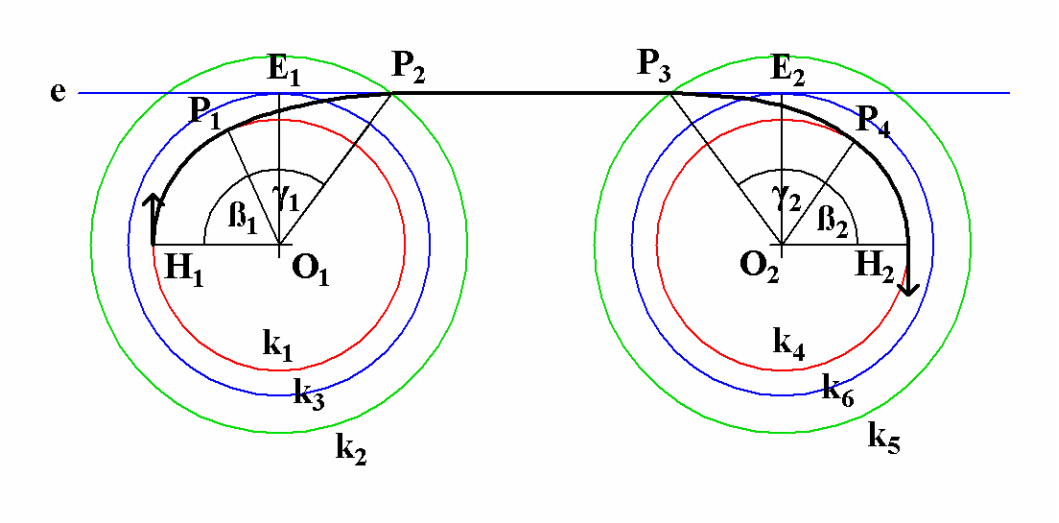
Tulajdonképpen elegendő meghatározni az E_1 és E_2 érintési pontokat. Ha ez megvan, meghatározható a többi szegmens határa is. A szegmensek kapcsolódnak, azaz egy szegmens végpontja a rákövetkező szegmens kezdőpontja lesz. A szegmenshatárokat kijelölő pontsorozat legyen P_0, P_1, P_2, P_3, P_4, P_5. Nyilvánvalóan $P_0 = H_1$ és $P_5 = H_2$.

A harmadik szegmens P_2 kezdő- és P_3 végpontja a k_2 illetve a k_5 kör és az e egyenes metszéspontja lesz. Itt vigyázni kell arra, hogy a két-két metszéspont közül a helyeseket válasszuk. Ha e szegmens határait ismerjük, kiszámíthatjuk a többiét is. Mivel a körök meghatározásánál már lejátszottuk a kormány kiegyenesítését, ismerhetjük a γ_1 szöget, mely megmutatja, hogy a jármű mekkora szöget fordul O_1 körül, mialatt kiegyenesíti a kormányt. Hasonlóan ismerhetjük γ_2-t, mely megadja, hogy a jármű mekkora szöget fordul O_2 körül, amíg befordítja a kormányt. E két szög ismeretében meghatározható P_1 és P_4. Ezek után megkapható az is, hogy az autónak mekkora szöget kell a fordulási középpontok körül fordulnia teljesen befordított kormánnyal az első és az utolsó szegmensben. Ezeket a szögeket jelölte fentebb β_1 és β_2.

Ily módon kiszámított pályatervekre mutatnak példát a következő ábrák.



16. ábra: Útvonalterv belső érintő esetén



17. ábra: Útvonalterv külső érintő esetén

A kiszámított pályaterv szerint az autó minden esetben valamelyik irányba teljesen befordított kormánnyal fog elindulni, és ugyanígy fog megérkezni a célba is, azaz $|\alpha_1| = |\alpha_2| = \delta$. Felvetődik a kérdés, hogy milyen pályatervet kapunk, amikor az első, közelítéses módszerrel működő algoritmus $|\alpha_1| < \delta$ induló kormányállást ír elő. Ekkor azon algoritmus működéséből fakadóan $\beta_1 = 0$, azaz a kormányt a jármű elindulása után azonnal el kell kezdeni kiegyenesíteni, tehát az első szegmens tulajdonképpen hiányzik. Természetesen hasonló eset előfordulhat az érkezési oldalon is, ekkor $\beta_2 = 0$. Szélsőséges esetben ez a helyzet akár mindkét oldalon egyszerre is előállhat.

Az imént felvázolt, geometriai számításokon alapuló algoritmust használva ilyen esetekben olyan pályatervet kapunk, ahol β_1 illetve β_2 kisebb ugyan, mint 360˚, de nagyon közel esik ahhoz. Ha β_1 ilyen, akkor ez azt jelenti, hogy mielőtt az autó elkezdené kiegyenesíteni a kormányt, csaknem egy teljes kört kell fordulnia konstans kormányállással O_1 körül. Ha pedig β_2-re kapunk ilyen értéket, akkor, mire a jármű teljesen befordítja a kormányát, éppenhogy túlcsúszik a célon, ezért, hogy mégis oda érkezzen meg, hasonlóan majdnem egy teljes kört kell fordulnia. Az ilyen terv nyilván előnytelen abból a szempontból, hogy a járműnek indokolatlanul hosszabb utat kell megtennie.

A megoldás az, ha azon az oldalon, ahol ilyen kedvezőtlen helyzet állt elő, kisebb mértékig befordított kormányállással tekintjük a járművet, majd ennek megfelelően újraszámoljuk a köröket, az érintőt, végül az egész pályatervet. Fontos látni, hogy ha csak

az egyik oldalon módosítottuk a kormányállást, akkor is újra kell számolni az egész pályát, mert, habár a másik oldalon lévő három kör középpontja és sugara nem változott, az ottani érintési pont el fog csúszni valamelyik irányba. Ez pedig akár azt is eredményezheti, hogy az újonnan kiszámított pályán ezen az oldalon is kedvezőtlen lesz a β szög értéke. Ennek akár a fordítottja is megtörténhet, azaz ha a két oldalon egyszerre áll elő ilyen helyzet, és csak az egyik oldali α-értéket változtatjuk, akkor akár olyan pályát is kaphatunk, amelynél már mindkét β-érték megfelelő. De tegyük fel, hogy ettől eltekintünk, és csak az egyik oldalra koncentrálva addig módosítjuk az ottani α értékét, amíg a kapott pályatervben az ottani β-érték megfelelő nem lesz. Könnyen lehetséges, hogy ebben a tervben a túloldali β-érték nem lesz megfelelő, ezért most csak erre az oldalra koncentrálva keressünk egy megfelelő α-értéket. Mire ezt megkapjuk, a másik oldali β újra elromolhat. Vagy, ha nem romlik el, akkor ott olyan helyzet áll elő, hogy $|\alpha| < \delta$, de $\beta \neq 0$, ami szintén előnytelen, hiszen azt írja elő, hogy a jármű tegyen meg bizonyos utat tartva a nem teljesen befordított kormány állását. Hogy ezt a helyzetet elkerüljük, ezen az oldalon ismét a kormány megfelelő irányba teljesen befordított állapotából indulva kellene folytatni a nekünk megfelelő pályaterv keresését, de legalábbis újra módosítani kellene az itteni α-t, ami megintcsak hatással lesz a túloldali szögekre.

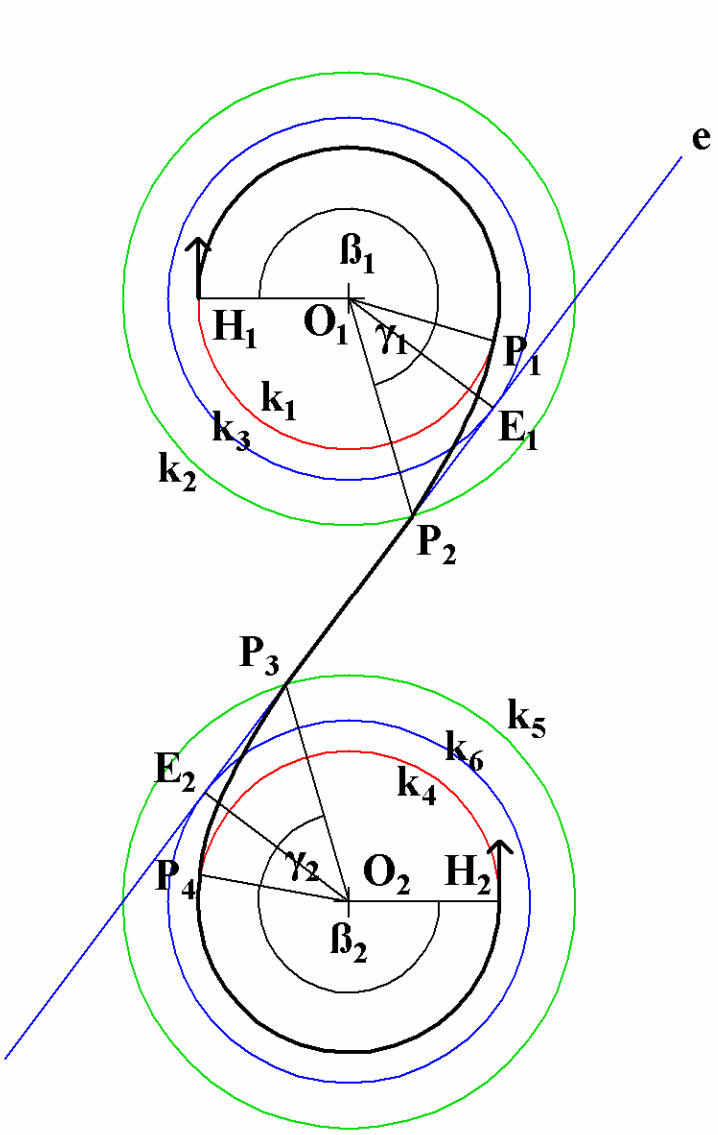
A problémát tehát az okozza, hogy a két oldal szögei nem függetlenek egymástól, ezért, még ha az előforduló problémákat másféleképpen is próbáljuk orvosolni, könnyen végtelen ciklusba kerülhetünk. Azonban, mivel a jármű valószínűleg úgysem fogja tudni pontosan tartani a pályatervet, nem fontos megtalálnunk az optimális α-értékeket, már az is elég, ha nem túl rosszak. Így megtehetjük, hogy csak bizonyos α-értékek mellett számítjuk ki a pályát, majd ezek közül a legmegfelelőbbet választjuk.

Az α-értékek lehetséges intervallumát célszerű *k* részre felosztani: ha az előzetes számítások alapján azt kaptuk, hogy $\alpha > 0$ kell legyen, akkor az $\frac{\alpha}{k}$, $\frac{2\alpha}{k}$, $\frac{3\alpha}{k}$, …, $\frac{(k-1)\alpha}{k}$, α értékek, $\alpha < 0$ esetén az $-\frac{\alpha}{k}$, $-\frac{2\alpha}{k}$, $-\frac{3\alpha}{k}$, …, $-\frac{(k-1)\alpha}{k}$, $-\alpha$ értékek mellett kell elvégezni a pályaszámítást. A két oldal lehetséges értékeinek minden kombinációjára el kell végezni a számítást, azaz összesen k^2-szer kell az aktuális paraméterekkel kiszámítani a pályát.

Ez a módszer még akkor sem támaszt jelentősebb számolásigényt az előbbi közelítéses módszernél, ha 1˚-os pontossággal szeretnénk megtalálni a legjobb útvonalat, de egyszerűbb és megbízhatóbb annál. A teljesítménye ugyanakkor növelhető például úgy, hogy *k*-t viszonylag kicsinek választjuk, majd ha ezzel a paraméterrel már meghatároztuk α_1 és α_2 értékét, azok $\frac{\delta}{k}$ nagyságú környezetét újra felosztva *k* részre pontosítjuk a megoldást. Ez a módszer a kívánt pontosság elérését biztosító mélységig folytatható.

Például δ=32˚ és 2˚-os elérni kívánt pontosság esetén az első módszerrel $16^2 = 256$ pályaszámítást kell elvégezni. A második módszerrel, például $k = 4$ paraméterrel számolva az első lépésben $4^2 = 16$ pályaszámítás után 8˚-os pontosságot érünk el, a második lépésben szintén 16 ilyen számítás után pedig elérjük a 2˚-os kívánt pontosságot. Azaz összesen 32 pályaszámítást végeztünk az alapmódszer 256-jával szemben, mégis elértük ugyanazt a pontosságot.

Hogy a fenti módszer megfelelően működjön, a programnak meg kell tudnia állapítani, hogy egy adott megoldás mennyire jó, illetve, hogy két megoldás közül melyik a jobb. Viszonylag könnyen számszerűsíthető egy erre alkalmas érték, ha tudjuk, hogy milyen tulajdonságú pályát szeretnénk kapni. Általános szabálynak tekinthető, hogy ha $|\alpha| \neq \delta$, akkor a neki megfelelő oldalon lévő β érték legyen nulla. Ha ez mégsem teljesül, akkor a pálya legyen annál kevésbé előnyös, minél nagyobb β abszolútértéke. Azt sem szeretnénk, ha a β-k közel 360˚-osak lennének. Bizonyos esetekben elkerülhetetlen, hogy a β-értékek nagyobbak legyenek, mint 180˚, de a közel teljes fordulást követelő 360˚-hoz közeli értékek elkerülhetők.



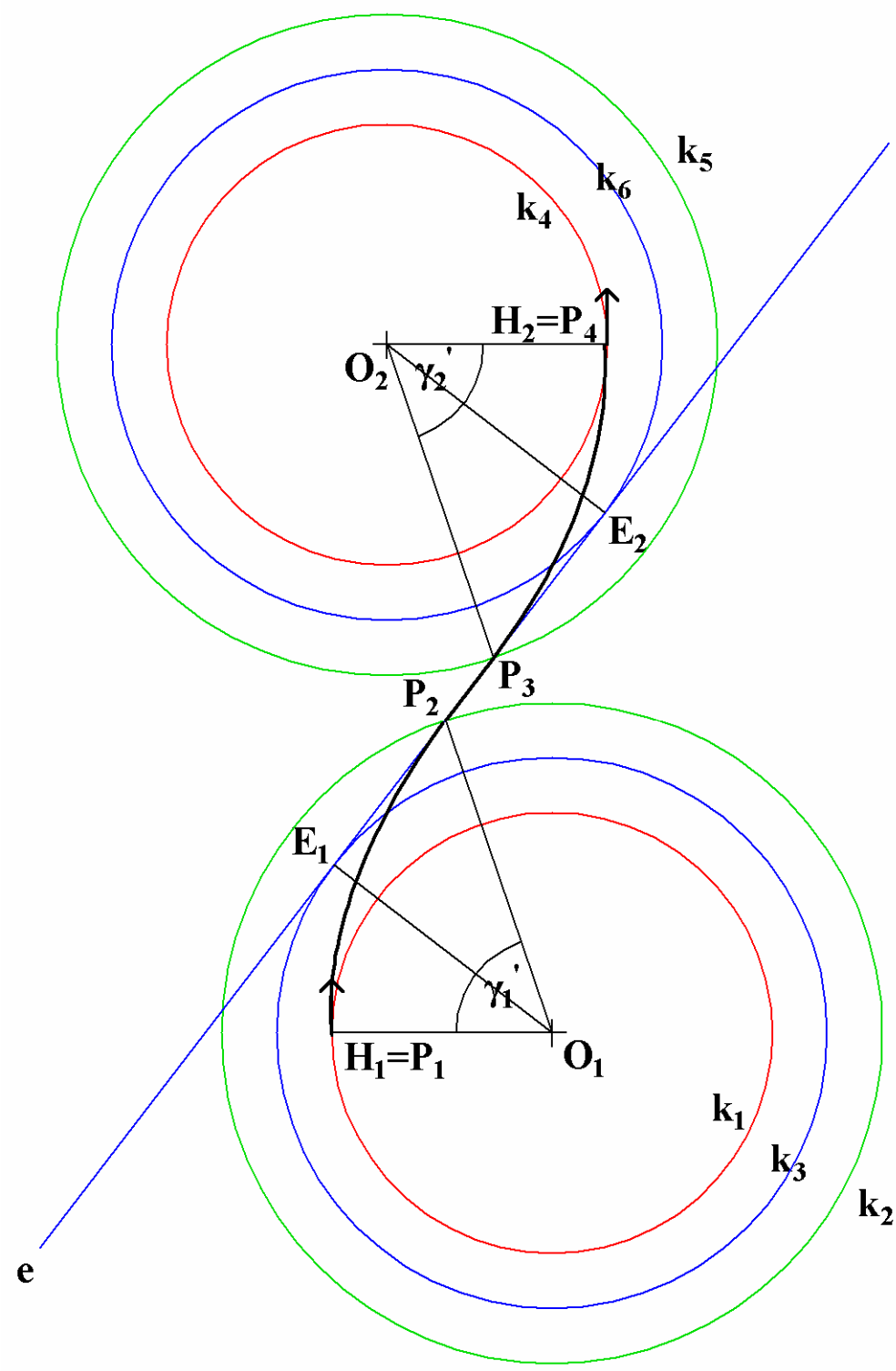
18. ábra: Egy példa arra az esetre, amikor a β-értékek mindenképp nagyobbak lesznek 180˚-nál

Amikor tehát nem indokolt a legalább 180˚-os β-érték, a vizsgált pálya annál előnyösebb, minél közelebb van a β a 0 vagy a 360˚-hoz. A 360˚-hoz képesti közelséget azért érdemes figyelni, mert szélsőséges esetben előfordulhat, hogy minden vizsgálandó α-értékhez ilyen 360˚-hoz közeli β-érték fog tartozni. Ilyen esetben megtehetjük, hogy a minél nagyobb β-értékeket részesítjük előnyben, az ellenőrzőpontok kijelölésekor azonban nem írjuk elő a csaknem teljes fordulást, hanem a β-t nullának tekintve az első vagy utolsó pályaszegmenst elhagyjuk. Az emiatt bekövetkező kisebb letérést a pályáról a jármű menet közben fogja korrigálni. Az indulási oldalon ez viszonylag könnyű, mert

ekkor még a jármű előtt áll egy hosszabb-rövidebb egyenes szakasz, aminek megtétele alatt bőven lesz lehetőség a korrekcióra. Az első szegmenst tehát elhagyjuk, az addigi második kezdőpontja pedig a jármű kiindulási pozíciója lesz. Az érkezési oldalon nem ez a helyzet, ezért itt mindenképp az a legjobb, ha a β értéke nem 360˚-hoz közeli. Ha ezt mégsem tudjuk elkerülni, itt is a 360˚-hoz minél közelebbi értékre fogunk törekedni, majd a pálya kijelölésekor az utolsó szegmenst elhagyjuk, az addigi utolsó előtti végpontja pedig az elérni kívánt célpozíció lesz. Mivel a β értéke elég közel fog esni a 360˚-hoz, az emiatt érzékelhető eltérés nem lesz nagyobb annál, amit a terep egyenetlenségei okoznak. Ezeken túl előnyben részesítjük a minél nagyobb abszolútértékű α-értékeket, mivel el szeretnénk kerülni az olyan pályát, ahol nem teljesen befordított kormánnyal kell a kormányállást tartva fordulni.

Előállítottuk tehát a pályatervet, amelyet alapesetben öt fő szegmens határoz meg, melyek közül akár több is hiányozhat. Figyelembe kell vennünk azonban bizonyos, a pályán való mozgás közben jelentkező tényezőt is.

A legfontosabb ilyen, hogy a következő ellenőrzőpontig el kell látnia a járműnek, hogy az esetlegesen szükséges korrekciót el tudja végezni. Ehhez elengedhetetlen a terep megfelelő részletének ismerete, melyet a jármű szenzorai térképeznek fel. Ezért az ellenőrzőpontokat akkora távolságra kell felvenni egymástól, hogy az aktuális mindig a jármű szenzorainak hatókörén belül legyen. Ez meghatározza az ellenőrzőpontok maximális távolságát, ami miatt szükség lehet a fő pályaszegmensek feldarabolására. Ennek lesz egy másik fontos következménye is: ha az ellenőrzőpontok közti távolság kisebb, kevesebbet kell előreszámolnunk a korrekció meghatározásához, ezáltal gyorsabb futást érhetünk el.



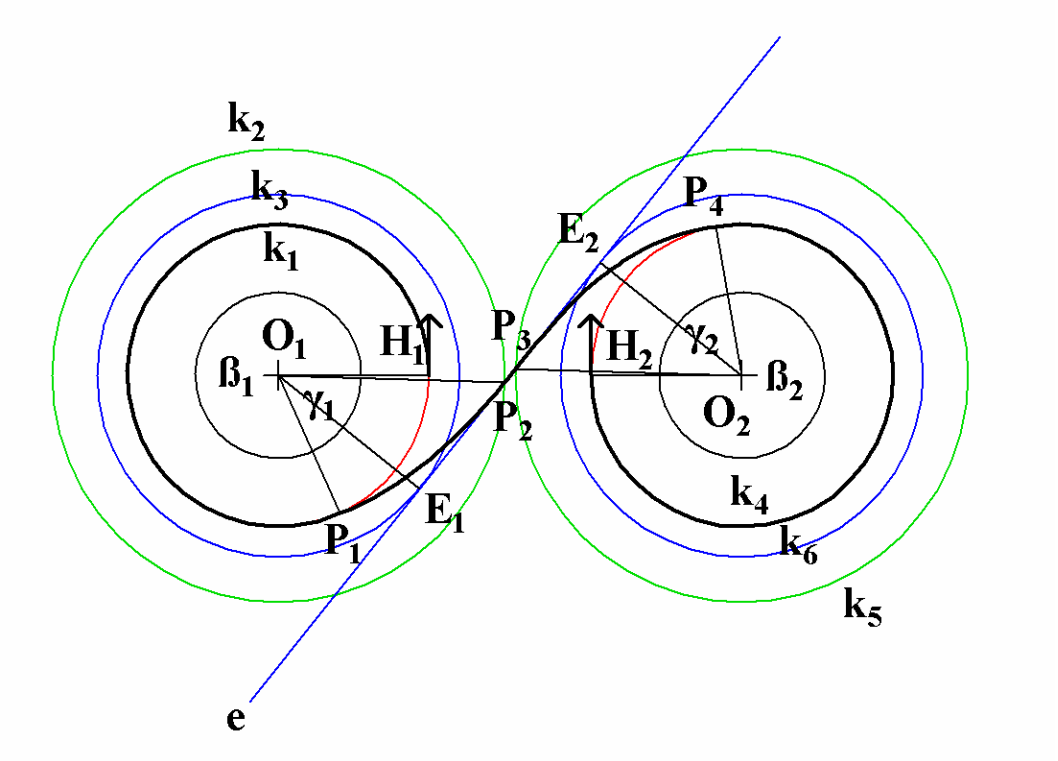
19. ábra: Egy példa arra az esetre, amikor a β-értékek nullák

Egy másik fontos dolog a pályakövetés módjából adódik. Az aktuális ellenőrzőpontig addig fogjuk előreszámolni a jármű helyzetét, amíg közeledik ahhoz. Amint távolodni

kezd tőle, befejezzük a számítást, és meghatározzuk a szükséges korrekciót. Abban az esetben, ha az első vagy az utolsó szegmens 180˚-nál nagyobb fordulást határoz meg, a jármű, mielőtt közeledni kezdene az ellenőrzőponthoz, valamelyest távolodni fog attól. Sőt, a pályáról való letérések miatt akár ennél valamivel kisebb szögű fordulások esetén is előfordulhat ilyen eset. Ekkor az adott szegmens által leírt fordulási ívet addig kell felezgetni, amíg a kisebb ívekhez tartozó pályaszegmensek 90˚-nál kisebb fordulást nem írnak elő. Így ugyanis biztosan nem kezd el távolodni a jármű a következő ellenőrzőponttól azelőtt, hogy valóban közel kerülne ahhoz.

A fentiek következtében a végleges pályatervet ötnél sokkal több szegmens is alkothatja, melyeket tehát a főszegmensek alkalmas feldarabolásával kaptunk. A szegmenseket a következő információkkal írhatjuk le: kezdő- és végpontjuk helyzete, továbbá a jármű kívánt állása, amikor áthalad azokon. Ismerni kell továbbá, hogy a jármű milyen kormányállással indul el.

Néhány kisebb változtatással elérhető, hogy a fenti algoritmussal azokban az esetekben is megtervezhető legyen egy pálya, amikor a cél viszonylag közel esik a jármű kiindulási helyzetéhez. Mindössze az indulási és az érkezési pozícióbeli kormányállások szögeinek előjeleit kell úgy módosítani, hogy megszerkeszthető legyen a pálya, azaz, ha belső érintőt kell számolnunk, akkor a k_3 és a k_6 körök ne metsszék egymást, és az egyenesen megteendő utat tartalmazó harmadik fő pályaszegmens elhelyezkedése is megfelelő legyen. Ily módon viszonylag kevés módosítás árán képessé tettük a járművet arra is, hogy viszonylag közeli célba is eljusson, igaz, viszonylag hosszú úton, mivel nem használtuk ki, hogy az autó tolatni is képes.



20. ábra: Egy példa arra, amikor a cél közel esik a kiindulási pozícióhoz

5.2 Igazodás a pályatervhez

A járművet a kiszámított útvonalon tartani ezek után már nem nehéz. Induláskor az első pályaszegmens végén lévő ellenőrzőpontot próbálja elérni, majd, ha ez többé-kevésbé már sikerült, a következőt, és így tovább. Az aktuálisan figyelt ellenőrzőpontot, azaz az aktuális szegmens végpontját jelölje *C*, a jármű kívánt állását, amikor *C*-n áthalad, jelölje l_C. Mint fentebb már említettem, a jármű aktuális pozíciójából kiindulva addig számolunk előre az aktuális kormányállást tartva, amíg a jármű távolodni nem kezd a *C* ponttól. Ez után már meg tudjuk mondani, hogy milyen korrekcióra van szükség: attól függően fordítjuk a kormányt, hogy a *C* pont a jármű melyik oldalán helyezkedik el ebben a pillanatban.

A jármű a pályaszámítás során meghatározott kezdeti kormányállást veszi fel indulás előtt, majd álló helyzetből folyamatosan gyorsítva indul el. Sebességét igyekszik a javasolt utazósebességhez igazítani.

Az aktuálisra rákövetkező ellenőrzőpontot akkor kezdi el figyelni, ha az aktuálistól már távolodik, vagy átlépi a C pont és az l_C irányvektor által meghatározott egyenesre merőleges, C-n áthaladó egyenest.

Ha azt tapasztalja, hogy jelentősen el fog térni az ellenőrzőponttól, visszavesz a sebességből, ezáltal a jármű előtt nagyobb korrigálási lehetőség nyílik. Elegendő, ha elveszi a gázt, fékeznie nem szükséges. Hogy a jármű mégse lassuljon le nagyon, célszerű például az utazósebesség feléig engedélyezni a lassulást. Ha az autó lejtőn halad, sebessége átlépheti a javasolt utazósebességet, ami az útvonal tartása miatt már indokolttá teheti a fékezést. Érdemes megadni egy korlátot, például a javasolt sebesség 110%-át, amit átlépve a jármű fékezni kezd. Ez még nem okoz túl nagy letérést a pályáról, ugyanakkor biztosít némi szabadságot a jármű számára, hogy ne kelljen folyamatosan használnia a pedálokat.

A célhoz közeledve a járműnek lassítania kell, majd a célt elérve meg kell állnia. A pillanatnyi sebességet és a fékek jellemzőit ismerve kiszámítható a fékút, ezáltal meghatározható, hogy mikor kell a járműnek megkezdenie a fékezést. Mivel az emberi vezető sem a fékpedált teljesen benyomva fékez, számolhatunk például a fékhatás ⅔ részével, ami hosszabb fékutat eredményez ugyan, de lehetővé teszi azt is, hogy az autó akár lejtőn is időben meg tudjon állni.

A tapasztalatok szerint azt mondhatjuk, hogy a jármű utazósebességét az ellenőrzőpontok távolságához, azaz végsősoron a szenzorok hatóköréhez kell igazítani. Nem jó, ha az ellenőrzőpontok sűrűn helyezkednek el. Ekkor a jármű a legkisebb letérésekre is azonnal agresszívan reagál, ami miatt a következő ellenőrzési pont elérése érdekében kicsit arrébb még nagyobb, ellentétes irányú korrekciót kell tennie. Ez akár addig is folytatódhat, amíg a kijelölt pályától való eltérés akkora nem lesz, hogy a jármű már nem tudja azt kezelni, és valahol a céltól távol egyszerűen megáll. Szintén gondot okozhat, ha az ellenőrzőpontok távol esnek egymástól. Ez a pályaterv jellege miatt szinte csak az egyenesen megteendő utat leíró szegmenseknél jelentkezik. Ekkor ugyanis a letérésre csak lassan reagál a jármű, azaz nem próbál minél gyorsabban visszatalálni a kijelölt egyenes útra, hanem az ellenőrzőpontot veszi célba. Emiatt ott nem a kívánt állással halad át, ami ismét letérést okozhat a nem egyenesen megteendő következő szegmensek megtétele során. Mivel ekkor már közel van a célhoz, a járműnek már csak viszonylag kevés lehetősége van korrigálni. Bár ezeken a hibákon a fentebb leírt

sebességcsökkentéssel sikerült javítani, a legbiztosabb módszer mégis az utazósebesség alkalmas megválasztása.

6 A program használata

Ahogy már esett róla szó, a program rendkívül jól paraméterezhető. Ha mást nem mondunk, a program a fő konfigurációs fájlt a futtatható állománnyal azonos szinten található *data* könyvtárban, *settings.init* néven keresi. Ha ez hiányzik, a program hibaüzenettel leáll. Parancssori argumentumként a *–init* kapcsoló után megadható ettől eltérő konfigurációs fájl is.

Ez a fő konfigurációs fájl tartalmaz néhány alapvető futási paramétert: az egyik meghatározza, hogy az alkalmazást ablakban vagy teljes képernyős módban kell-e futtatni, emellett megadandóak az ablakméretet, a felbontást és a színmélységet meghatározó paraméterek is. Ugyan az alkalmazás fő célja a jármű felhasználói beavatkozást nem igénylő eljuttatása a kijelölt pozícióba, meghagytam a lehetőséget, hogy a felhasználó billentyűzetről irányíthassa a járművet. Ebben az esetben nem kerül sor a pályaszámításra sem, a felhasználó szabadon mozoghat a járművel a virtuális világban. Ezen lehetőség kihasználására szintén egy paraméter biztosít lehetőséget.

Két fontos paraméter a járművet és a terepet leíró konfigurációs fájl neve. A jármű esetében megadható a beépített autótípusok neve is, ekkor az autó adatait a program nem a konfigurációs fájlból tölti be, hanem lefuttatja az ezeket beállító metódus, ezáltal megspórolva egy fájl beolvasását. Mindazonáltal a program nincs korlátozva az általam elkészített autómodellekre, hanem némi munkával tetszőleges modell alkalmassá tehető a programban történő felhasználásra. Mindössze a karosszéria és a jobb első kerék modelljét kell az irányoknak megfelelően letárolni egy-egy *3ds* kiterjesztésű fájlban, majd elkészíteni az autó konfigurációs fájlját.

A fentieken túl még mindig a fő konfigurációs állományban adandó meg néhány olyan paraméter, ami sem a járműhöz, sem a terephez nem köthető. Ilyenek a kezelőfelület és az ég paraméterei és textúrái.

Végül a jármű automatikus vezérléséhez elengedhetetlen paraméterek következnek: a célpozíció és az ottani állás, a pálya megtervezéséhez szükséges paraméterek, így a fentebb említett k érték és a számítás mélységét megadó konstans, a számítás közben használt sebesség- és Δt-érték, továbbá az ellenőrzőpontok maximális távolsága és a javasolt utazósebesség is.

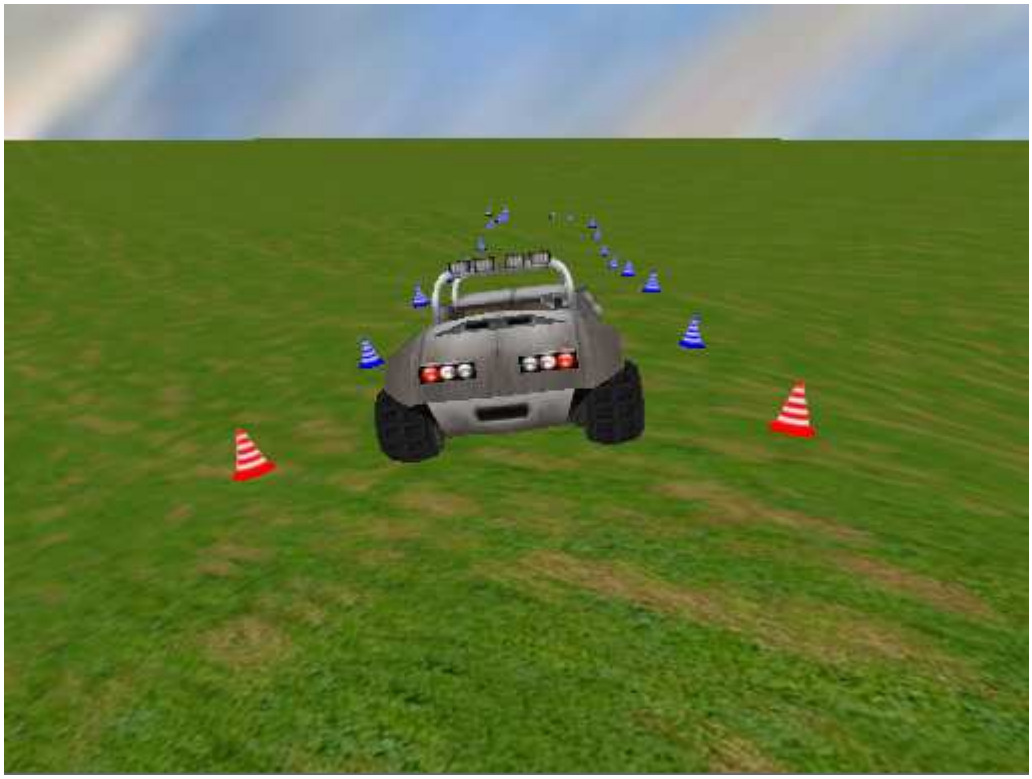
A terep és a jármű konfigurációs fájljairól a megfelelő fejezetekben már esett szó. Mindhárom konfigurációs állomány szöveges a könnyű olvashatóság és szerkeszthetőség miatt.

Futás közben, amennyiben a felhasználó irányítja a járművet, a nyíl-, vagy a W, A, S és D billentyűkkel vezérelheti a pedálokat illetve a kormányt. A kezelőfelületen látható az első kerekek, a kormány és a pedálok állása, valamint a pillanatnyi sebesség és az egyes kerekekre eső terhelések aránya is. Mindezek természetesen láthatók automatikus vezérlés közben is. Ekkor a kiszámított pályát a szegmensek határán megjelenő bóják jelölik. A szimuláció a szóköz billentyű lenyomásával szüneteltethető, illetve folytatható.

Háromféle kameramódban szemlélhetjük az eseményeket. Az első a V billentyű lenyomásával aktiválható, ezután a kamera a virtuális világban szabadon mozgatható az I, J, K, L, U és O billentyűkkel. A második lehetőség az autót követő kameraállásokat jelenti, azaz amikor a kamera relatív helyzete a járműhöz képest állandó. Ez a mód a C billentyűvel aktiválható, illetve ezzel válthatunk a különböző nézetek között. Az autót szemlélhetjük oldalról, elölről, hátulról vagy akár felülről is, követve annak mozgását. A harmadik módban is az autót fogja követni a kamera, de pozíciója mindig az autót és a terep valamelyik sarkát összekötő egyenesen lesz. Ez a mód a B billentyűvel aktiválható, illetve ezzel lehet váltani a terep sarkai között. A kamera magasságát mindhárom esetben a * és a / billentyűkkel szabályozhatjuk, míg az utóbbi két mód esetén a kamerának az autótól mért távolságát a + és a – billentyűkkel állíthatjuk be. A program a kis- és nagybetűk között nem tesz különbséget.

A funkcióbillentyűkkel néhány egyéb lehetőség is elérhető. Az F1-gyel bekapcsolható a kerekek mellett megjelenő, az adott kerékre eső terhelés nagyságát szimbolizáló szakasz. Az F2-vel elérhetjük, hogy a jármű karosszériája ne rajzolódjon ki, csak a kerekek. Az F3-mal a terep textúrázott illetve csak a drótvázmodellt ábrázoló megjelenítési módjai között válthatunk. Az F4-gyel kikapcsolható a LOD-technika alkalmazása, azaz így elérhető, hogy az egész terep teljes részletességgel jelenjen meg. Az F5-tel kikapcsolható a kezelőfelület.

21. ábra: Képernyőkép a szimulációból (az autót a felhasználó vezérli)



22. ábra: Képernyőkép a szimulációból automatikus vezérlés közben

7 Összefoglalás és fejlesztési lehetőségek

A kitűzött célokat illetően elmondható, hogy a jármű mozgása kellően valósághű, illetve elfogadható pontossággal képes a kijelölt pozícióra eljutni. Született egy olyan járműdinamikai modell, melynek elkészítése során leginkább a kormányzás és a felfüggesztési rendszer valósághű viselkedésére figyeltem, ugyanakkor a továbbfejleszthetőséget is szem előtt tartottam. Hasonlóan nagy odafigyelést igényelt a pályatervet elkészítő kódrészlet megírása, melynek során több új ötlet is felmerült, emiatt kétségtelenül ez a kód legtöbb átdolgozáson átesett része. Mindezt egy megfelelő mértékben valósághű grafikus megjelenítés teszi látványossá.

Fejlesztési lehetőségek azonban minden területen akadnak, sőt, talán csak a képzelet és a számítókapacitás szabhat nekik határt.

A terepmodellt elsősorban a LOD-technika csiszolásával kellene javítani, mely egyelőre elég kezdetleges, de a későbbiekben egy összetettebb terepmodell bevezetése is indokolttá válhat. Emellett az eddigi sivárabb környezetet növényzettel és tereptárgyakkal lehetne gazdagítani.

A járműdinamikai modell valósághűségét elsősorban a dinamikus erők kezelésének bevezetésével lehetne fokozni. Ilyenek a jármű kanyarodása és gyorsítása vagy lassítása során fellépő oldal- és hosszanti irányú erők. Modellezni kellene a megcsúszásokat is, ezzel együtt bevezetni a talaj tapadási együtthatóját, mely a terep különböző részein akár eltérő is lehet. Ezzel párhuzamosan le kellene írni egy összetett abroncsmodellt, és kezelni kellene a szél erősségét és irányát is. Ezek mellett le kellene programozni a jármű további alkatrészeinek a működését, illetve a meglévőket tökéletesíteni. A motor jelenleg folyamatosan konstans nyomatékot ad le, ehelyett egy fordulatszámtól függő nyomatékgörbét kellene bevezetni. Ezzel párhuzamosan szimulálni kellene a sebességváltó és a kuplung működését is. A felfüggesztés geometriája meglehetősen egyszerű, amit érdemes lenne lecserélni egy olyanra, ami közelebb áll a valóságoshoz. Egy további izgalmas lehetőség a kézifék megvalósítása.

A pálya megtervezése és követése terén szintén sok fejlesztési lehetőség nyílik. Elsősorban a viszonylag közeli célba vezető utak megtervezését kellene tökéletesíteni a jármű tolatási képességének kihasználásával. Az útvonalkövetést sokkal pontosabbá lehetne tenni, ha a jármű több ellenőrzőponttal előrébb tekintene, még ha az elsőn túli

terepviszonyokat nem is ismerheti. A feladatot sokkal izgalmasabbá teheti, ha a virtuális világban akadályok is vannak, melyeket figyelembe kell venni a számításoknál, hiszen a járműnek ki kell kerülnie azokat.

Mindezek mellett természetesen a grafikus rendszer is jelentős továbbfejlesztési lehetőségeket rejt. Részecskerendszerek segítségével megvalósítható a kerekek által felvert por és a füst. Érdemes lenne definiálni egy megvilágítási rendszert. A valósághűség fokozható a mozgás miatti elmosódás kezelésével, valamint a környezet és az autó hangjainak megszólaltatásával. A látványosság nagymértékben fokozható lenne az ütközésdetektálás és a karosszéria deformálódásának megvalósításával.

8 Köszönetnyilvánítás

Köszönetet szeretnék mondani témavezetőmnek, dr. Herendi Tamásnak, aki ötleteivel és tanácsaival nagyban segítette munkámat.

Köszönettel tartozom továbbá Szüleimnek, akik nem csak ezen munkám során, hanem tanulmányaim alatt végig támogattak.

9 Irodalomjegyzék

[1] Benedek Zoltán, Levendovszky Tihamér: Szoftverfejlesztés C++ nyelven
SZAK Kiadó Kft., 2007

[2] http://www.cplusplus.com/

[3] Szirmay-Kalos László, Antal György, Csonka Ferenc: Háromdimenziós grafika, animáció és játékfejlesztés
ComputerBooks, Budapest, 2006

[4] OpenGL Architecture Review Board, Mason Woo, Jackie Neider, Tom Davis, Paula Wom: OpenGL Programming Guide – The Official Guide to Learning OpenGL
Addison-Wesley Publishing Company, 1997

[5] http://www.opengl.org/

[6] http://nehe.gamedev.net/

[7] http://www.opengl.org/documentation/specs/glut/spec3/spec3.html

[8] http://www.vterrain.org/

[9] Deák Szabolcs: Dynamic Simulation in a Driving Simulator Game
Budapesti Műszaki és Gazdaságtudományi Egyetem

[10] Konyha Zoltán: Aspects of Developing a Driving Simulation Game
Budapesti Műszaki és Gazdaságtudományi Egyetem