VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Simulace pohybu auta

Ročníkový projekt

2006 Tomáš Dvořák

Zadání

Název: Simulace pohybu auta

Vedoucí: Pečiva Jan, Ing., UPGM FIT VUT

Zadání:

1. Nastudujte si problematiku rendrování virtuálních scén.
2. Navrhněte algoritmy pro fyzikální model a pohyb auta.
3. Navržené algoritmy implementujte a demonstrujte na jednoduché aplikaci.
4. Vyhodnoťte vizuální výsledky a realističnost výsledné simulace.
5. Práci publikujte na internetu.

Kategorie: Počítačová grafika

Implementační jazyk: C++

Volně šířený software: Coin

Literatura:

- Open Inventor tutoriál na ROOT.CZ
- Josie Wernecke, The Inventor Mentor, Addison-Wesley Professional, 1994, ISBN: 0201624958

Prohlášení

Prohlašuji, že jsem tento ročníkový projekt vypracoval samostatně pod vedením pana Ing. Jana Pečivy.

Uvedl jsem všechny prameny a materiály, ze kterých jsem čerpal.

V Brně dne

Tomáš Dvořák

Abstrakt

Tento projekt se zabývá simulací pohybu auta ve 3D scéně. Snaží se o přiblížení ovládání z klávesnice skutečnému ovládání auta pomocí volantu a pedálů. Pohyb ve scéně je založen na parametrech modelu auta, které jsou nastaveny podle skutečného vozidla.

Součástí projektu je ukázková aplikace, která demonstruje ovládání a pohyb auta ve scéně s překážkami. Aplikace je implementována v jazyce C++ s použitím knihovny Coin3D.

Klíčová slova

3D scéna, OpenGL, Open Inventor, Coin3D, fyzika, simulace, translace, rotace, detekce kolizí.

Abstract

This project deals with a car movement simulation in a 3D scene. It tries to aproximate keyboard control to real car control through the use of steering wheel and pedals. The movent in the scene is based on a car model parameters that are adjusted in accordance with a real vehicle.

One part of this project is a demo application which demonstrates the control and the car movement in a scene with obstructions. This application is implemented in C++ with a usage of the Coin3D library.

Keywords

3D scene, OpenGL, Open Inventor, Coin3D, physics, simulation, translation, rotation, collision detection.

Obsah

Obsah 6

1 Úvod 7

2 Aplikační prostředí 8
2.1 Open Inventor 8
2.2 Coin3D 9

3 Simulace 10
3.1 Model auta 10
3.2 Ovládání 12
3.3 Translace 13
3.4 Rotace 15
3.5 Detekce kolizí 16

4 Demonstrační aplikace 18
4.1 Popis aplikace 18
4.2 Ovládání aplikace 19

5 Závěr 20
5.1 Zhodnocení 20
5.2 Pokračování projektu 20
5.3 Internetová prezentace 20

Literatura 21

1 Úvod

Tento projekt se zabývá simulací pohybu auta po dvourozměrném terénu. Důraz je kladen na to, aby pohyb auta probíhal na základě parametrů, jejichž hodnoty vycházejí z hodnot u skutečného auta.

V následující kapitole je popis aplikačního prostředí a použitých knihoven.

V kapitole 3 je pak popsána hlavní část projektu, tedy simulace pohybu auta.

Další kapitola se zmiňuje o ukázkové aplikace, ve které jsou použity algoritmy z kapitoly 3.

Poslední kapitolou je kapitola Závěr, ve které je zhodnocení dosažených výsledků. Jsou zde také nastíněna možná pokračování projektu. Nakonec je zde uvedena internetová adresa, na které je tento projekt umístěn k nahlédnutí, popř. ke stažení.

2 Aplikační prostředí

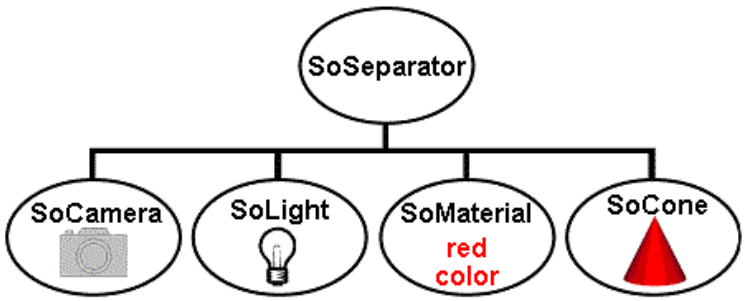
Vývoj projektu probíhal pod operačním systémem Windows a také demonstrační aplikace je určena k běhu pod tímto systémem. Při vývoji jsem použil vývojové prostředí Microsoft Visual Studio 2003. Aplikace je napsána v programovacím jazyce C++ s využitím knihovny Coin3D [1], která je kompatibilní s Open Inventor API. Při vývoji jsem využíval dokumentaci ke této knihovně [2], knihu The Inventor Mentor [3] a také tutoriál [4].

2.1 Open Inventor

Open Inventor je knihovna napsaná v C++. Je postavena nad OpenGL. Nahrazuje primitivní rozhraní OpenGL množinou tříd, které usnadňují práci programátorovi při vývoji aplikací. Aplikace psané pomocí Open Inventoru mohou být také výkonnější než aplikace psané přímo pomocí OpenGL, protože Inventor může provádět optimalizace nad daty scény.

Design Inventoru vychází z konceptu grafu scény. Scéna je složena z uzlů, které jsou uspořádány do stromové struktury. Uzly mohou být různých typů, některé nesou informaci o geometrii, jiné o transformaci nebo o atributech objektu. Významné jsou také uzly, které seskupují jiné uzly do podstromů. Díky podstromům je pak snadné provádět transformace nad částmi scény. Uzel s transformací v podstromu neovlivňuje transformaci objektů, které jsou ve stromu scény blíže ke kořenu.

Jednoduchý strom scény je na obrázku 2.1.

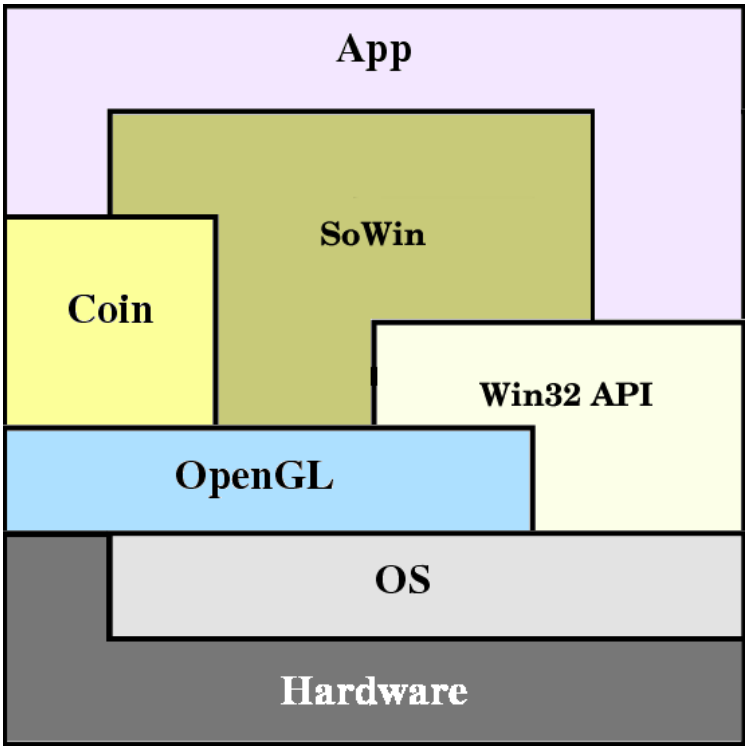


obrázek 2.1

Informace v této kapitole uvedené jsem čerpal z tutoriálu [4].

2.2 Coin3D

Jak je zmíněno výše, Coin 3D je knihovna kompatibilní s Open Inventorem. Jádro Coin je platformě nezávislé, může běžet na platformě MS Windows, Linux, Unix nebo SGI Irix. Vrstvu mezi Coin a aplikačním rozhraním operačního systému tvoří pomocné knihovny. Pro systém Windows je to knihovna SoWin. Na obrázku 2.2 jsou znázorněny propojení grafických knihoven a systémových součástí.



obrázek 2.2 z [1]

3 Simulace

V této kapitole jsou popsány simulační algoritmy a další postupy použité při simulaci pohybu auta.

Vlastní simulace probíhá v časových krocích daných konstantou *TICK*, která udává délku jednoho kroku v sekundách. Velikost této konstanty je nutné volit přiměřeně malou, aby simulace vypadala plynule. Já sem zvolil hodnotu 0.01, která odpovídá frekvenci zobrazování 100Hz. Při této frekvenci je již simulace dokonale plynulá.

Pro načasování jednotlivých kroků je použit časovač, který je na staven na hodnotu konstanty *TICK*. Tento časovač pravidelně spouští svou *callback* funkci. V této funkci pak probíhá výpočet parametrů scény při simulaci.

V každém kroku se provádějí následující operace:

- zjištění stisknutých kláves, popsáno v podkapitole 3.2
- výpočet přírůstků posunutí auta ve scéně, více v podkapitole 3.3
- výpočet přírůstku otočení auta ve scéně, podkapitola 3.4
- vypočtení nových souřadnic umístění modelu auta ve scéně na základě přírůstků posunutí a otočení
- detekce kolizí s překážkami, popis v podkapitole 3.5
- pokud došlo ke kolizi, návrat k předešlým souřadnicím, na kterých ke kolizi nedocházelo

Pro veškeré simulační výpočty jsou použity parametry modelu popsané v následující podkapitole 3.1.

3.1 Model auta

Při popisu modelu auta jsem použil konstantní parametry, jejichž hodnoty vycházejí z vlastností skutečného auta. Parametry bych rozdělil do dvou skupin: výkonové a dispoziční. Výkonové parametry jsou použity při výpočtu posunutí modelu ve scéně. Jsou to:

- *m* – hmotnost auta v [kg]
- *Pt* – maximální výkon motoru v [kW]
- *Pb* – maximální výkon brzdného systému auta v [kW]
- *Cx* – součinitel odporu

Dispoziční parametry jsou důležité pro výpočet rotace modelu a pro zjištění kolizí s překážkami. Jsou to:

- *MaxSteer* – hodnota maximálního natočení kol od přímého směru ve [°]
- *Width* – šířka auta v [m]
- *Length* – délka auta v [m]
- *AxleBase* – rozvor (vzdálenost náprav) v [m]

Model je dále popsán třemi parametry, jejichž hodnota je stanovena spíše odhadem, než podle skutečného auta. Jsou to parametry použité při ovládání :

- *maxThrottleTime* – čas potřebný pro sešlápnutí plynového pedálu od nulové polohy do maxima v [s]
- *maxBreakTime* – čas potřebný pro sešlápnutí brzdového pedálu od nulové polohy do maxima v [s]
- *maxSteerTime* – čas potřebný pro otočení volantu od přímé polohy do maxima na jednu stranu v [s]

Stejný čas platí i pro opačný pohyb ovládacích prvků auta, tedy např. návrat plynového pedálu zpět z maximální polohy do nulové.

Aktuální stav ovládacích prvků auta, aktuální pozice modelu ve scéně a hodnoty fyzikálních veličin jsou uloženy v následujících proměnných:

- *Throttle* – procento sešlápnutí plynového pedálu v intervalu <0,0; 1,0>
- *Break* – procento sešlápnutí brzdového pedálu v intervalu <0,0; 1,0>
- *Steer* – procento otočení volantu v intervalu <-1,0; 1,0>
- *xpos* – pozice auta ve scéně ve směru osy x
- *zpos* – pozice auta ve scéně ve směru osy z
- *angle* – otočení auta ve scéně vzhledem ke kladnému směru osy z
- *Speed* – aktuální rychlost auta

3.2 Ovládání

V tomto projektu jsem se snažil přiblížit ovládání modelu ovládání skutečného auta. Proto jsou v modelu použity proměnné *Throttle, Break* a *Steer.* Tyto proměnné reprezentují aktuální stav ovládacích prvků auta. Pomocí jejich hodnot se počítá aktuální výkon motoru, výkon brzdného systému a úhel natočení kol. Změna hodnot se provádí na základě vstupů z klávesnice. Neměl by však být problém měnit tyto hodnoty např. pomocí autopilota, který by auto řídil na základě svého algoritmu řízení.

Ošetření událostí stisku kláves je realizováno v souboru *Klávesnice.h.* Tento soubor pochází z jiného projektu [5], který je licencován jako Public Domain. Tato licence umožňuje volné použití částí díla a jejich úpravu (více na [6]).

Ovládání pomocí klávesnice probíhá pro jednotlivé ovládací prvky následovně:

- *Throttle* – je-li v aktuálním kroku simulace stisknuta šipka nahoru a je-li hodnota proměnné menší než 1, pak je hodnota *Throttle* zvýšena o podíl *TICK/maxThrottleTime.* Pokud šipka není stisknuta a hodnota je větší než 0, je hodnota proměnné o tento podíl snížena. Tím se docílí toho, že pokud je klávesa stisknuta po dobu *maxThrottleTime*, je hodnota *Throttle* rovna 1. Když po tuto dobu není stisknuta, je hodnota 0.
- *Break* – podobně jako u *Throttle*, pouze se počítá s konstantou *maxBreakTime* namísto *maxThrottleTime.*
- *Steer* – je-li v aktuálním kroku simulace stisknuta šipka vpravo a je-li hodnota proměnné menší než 1, pak je hodnota zvýšena o podíl *TICK/maxSteerTime.* Pokud šipka není stisknuta, neděje se nic. Je-li stisknuta šipka vlevo a je-li hodnota větší než –1, pak je hodnota o podíl snížena.

Chování je inspirováno chováním pedálů a volantu ve skutečném autě. Stejně jako v autě, pokud nemá řidič nohu na pedálu, se pedál vrací do výchozí polohy. Zde v simulaci, pokud není stisknuta šipka, se hodnota proměnné postupně vrací k nule. Polohy obou pedálů jsou nezávislé jak v autě, tak v této simulaci.

Chování volantu je rozdílné. V autě pokud řidič sundá ruce z volantu, volant zůstává v konstantní poloze. Stejně tak v této simulaci, pokud není stisknuta šipka vpravo či vlevo, zůstává hodnota proměnné neměnná.

3.3 Translace

Výpočet posunutí modelu ve scéně se provádí na základě výkonových parametrů z podkapitoly 3.1 a na základě aktuálních hodnot proměnných reprezentujících stavy ovládacích prvků auta.

Základní myšlenka pohybu je taková, že se skládá z mnoha krátkých rovnoměrných pohybů. Tyto pohyby probíhají jakoby mezi voláními *callback* funkce časovače (viz. Začátek kapitoly 3) a při každém volání této funkce se mění parametr pohybu rychlost.

Pro výpočet změny rychlosti je třeba vycházet ze vztahů pro pohyb rovnoměrně zrychlený. S výpočtem jsem začal od výkonu. Výkon je definován jako podíl práce a času, po který byla práce vykonávána

$$P = \frac{W}{t} \ . \tag{3.1}$$

Potom pro práci platí

$$W = P * t \ . \tag{3.2}$$

V simulaci koná práci jak motor, tak brzdný systém. Pro tyto práce platí vztahy

$$Wt = (Pt * Throttle) * TICK \ , \quad Wb = (Pb * Break) * TICK \ .$$

V těchto vztazích je vždy maximální výkon násoben hodnotou proměnné, která reprezentuje aktuální stav ovladače daného výkonu. Odečtením těchto prací získáme výslednou práci v simulaci $W = Wt - Wb$.

Práce je definována jako dráhový účinek síly

$$W = \vec{F} * s * \cos a \ . \tag{3.3}$$

V simulaci působí síly pouze rovnoběžně s podložkou, proto je *cos* a roven 1.

Pro dráhu zrychleného pohybu platí vztah

$$s = \frac{1}{2} * a * t^2 \ . \tag{3.4}$$

Pro sílu platí

$$F = m * a \ . \tag{3.5}$$

Ze vztahů 3.3, 3.4 a 3.5 dostaneme vztah pro zrychlení

$$a = \sqrt{\frac{2 * |W|}{m * t^2}} \ . \tag{3.6}$$

Toto zrychlení je maximální možné při velikosti proměnných *Throttle* a *Break.* Proti směru pohybu auta však při simulaci působí odporová síla prostředí, pro kterou platí vztah

$$F_{odp} = C_x * \frac{r}{2} * v^2 \ , \tag{3.7}$$

kde *Cx* je součinitel odporu, r je hustota prostředí a *v* je aktuální rychlost. Skutečné zrychlení se spočítá podle vztahu

$$a = \frac{(F - F_{odp})}{m} . \tag{3.8}$$

Pro zrychlení platí definiční vztah

$$a = \frac{\Delta v}{\Delta t} \ . \tag{3.9}$$

Podle vztahu 3.9 se pak v simulaci po dosazení *TICK* za Dt spočítá přírůstek rychlosti, který se přičte k aktuální rychlosti *Speed.*

Při rovnoměrném pohyb platí vztah pro dráhu

$$s = v * t \ . \tag{3.10}$$

V simulaci je tato rovnice použita ve tvaru *dist=Speed*TICK.* V proměnné *dist* je tedy přírůstek posunutí auta ve scéně. Tento přírůstek pak ještě musí být přepočítán do směrů os x a z, aby mohla být vypočtená nová pozice modelu ve scéně.
Pro přírůstky ve směru os platí

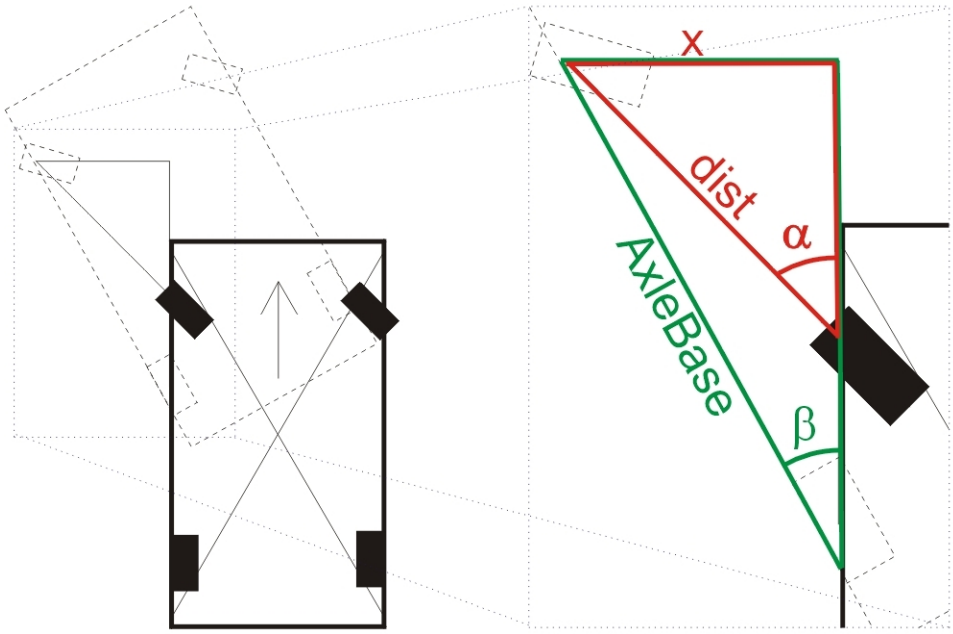
$$\Delta xpos = \sin(angle) * dist \ , \ \Delta zpos = \cos(angle) * dist \ .$$

Výsledná pozice ve scéně se získá přičtením těchto přírůstků a aktuálním souřadnicím.

Vztahy pro fyzikální veličiny v této kapitole jsem čerpal z knihy [7].

3.4 Rotace

V počátcích vývoje projektu jsem prováděl rotaci modelu auta o úhel natočení kol (na obr. 3.1 jde o úhel a). Velikost tohoto úhlu se spočítá jako *Steer*MaxSteer* (viz. Podkapitoly 3.1 a 3.2). To však neodpovídá skutečnému chování reálného auta. Celé auto se neotáčí o úhel natočení kol. Po zamyšlení nad zatáčením reálného auta jsem vytvořil metodu, která je graficky znázorněna na obr. 3.1.



obrázek 3.1

V levé části obrázku je pohled na celkovou situaci. Aktualní pozice auta je nakreslena plnou čarou, pozice po zatočení čarou čárkovanou. V pravé části obrázku je pak výřez oblasti, která je klíčová pro výpočet úhlu rotace auta.

Výpočet úhlu rotace auta (na obr. 3.1 úhel b) se provede na základě dvou aktuálních hodnot v simulaci a jedné konstanty:

- úhlu natočení kol a,
- vzdálenosti *dist* , výpočet viz. Podkapitola 3.3
- konstanty *AxleBase*, viz. Podkapitola 3.1.

V červeném trojúhelníku pro úhel α platí vztah

$$\sin a = \frac{x}{dist}. \qquad (3.11)$$

V zeleném trojúhelníku platí podobně pro b vztah

$$\sin b = \frac{x}{AxleBase}. \qquad (3.12)$$

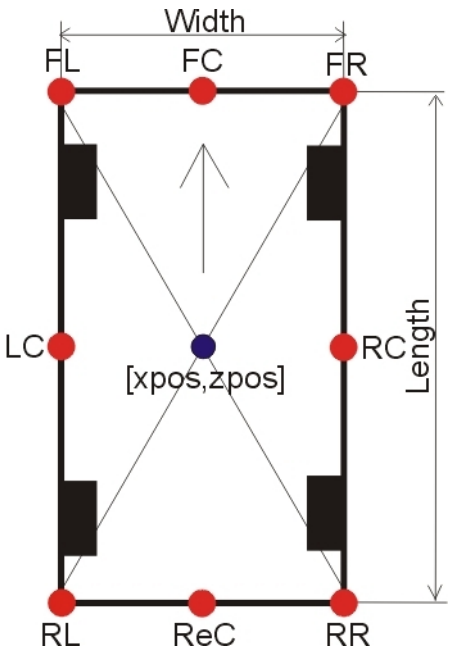
Dosazením *x* z rovnice 3.11 do rovnice 3.12 získáme výsledný vztah pro velikost úhlu otočení auta b

$$b = \arcsin \frac{\sin a * dist}{AxleBase}. \qquad (3.3)$$

Tento vypočtený úhel b však není výsledným úhlem otočení auta ve scéně. Je to pouze přírůstek tohoto celkového úhlu v tomto kroku simulace. Pro získání výsledného úhlu je tedy nutné tento přírůstek přičíst k aktuální hodnotě výsledného úhlu *angle*.

3.5 Detekce kolizí

Pro detekce kolizí se používají proměnné určující pozici auta ve scéně *xpos* a *zpos*, úhel natočení auta *angle* ve scéně a šířka a dálka modelu *Width* a *Length.* Kolize jsou detekovány na obvodu modelu v bodech, které jsou na obrázku 3.2 zobrazeny červeně.



obrázek 3.2

Souřadnice rohových bodů se spočítají podle vztahů:

- FrontLeft:

  FLx = *xpos*+(*Width*/2)*cos(*Angle*)+(*Length*/2)*sin(*angle*)

  FLz = *zpos*-(*Width*/2)*cos(*Angle*)+(*Length*/2)*sin(*angle*)

- FrontRight:
  FRx = *xpos*-(*Width*/2)*cos(*Angle*)+(*Length*/2)*sin(*angle*)
  FRz = *zpos*+(*Width*/2)*cos(*Angle*)+(*Length*/2)*sin(*angle*)
- RearLeft:
  RLx = *xpos*+(*Width*/2)*cos(*Angle*)-(*Length*/2)*sin(*angle*)
  RLz = *zpos*-(*Width*/2)*cos(*Angle*)-(*Length*/2)*sin(*angle*)
- RearRight:
  RRx = *xpos*-(*Width*/2)*cos(*Angle*)-(*Length*/2)*sin(*angle*)
  RRz = *zpos*+(*Width*/2)*cos(*Angle*)-(*Length*/2)*sin(*angle*)

Souřadnice bodů ve středu stran modelu se spočítají pomocí již spočítaných rohových bodů:

- FrontCenter:
  FCx = FLx-((FLx-FRx)/2)
  FCz = FLz-((FLz-FRz)/2)
- LeftCenter:
  LCx = FLx-((FLx-RLx)/2)
  LCz = FLz-((FLz-RLz)/2)
- RightCenter:
  RCx = FRx-((FRx-RRx)/2)
  RCz = FRz-((FRz-RRz)/2)
- RearCenter:
  ReCx = RRx-((RRx-RLx)/2)
  ReCz = RRz-((RRz-RLz)/2)

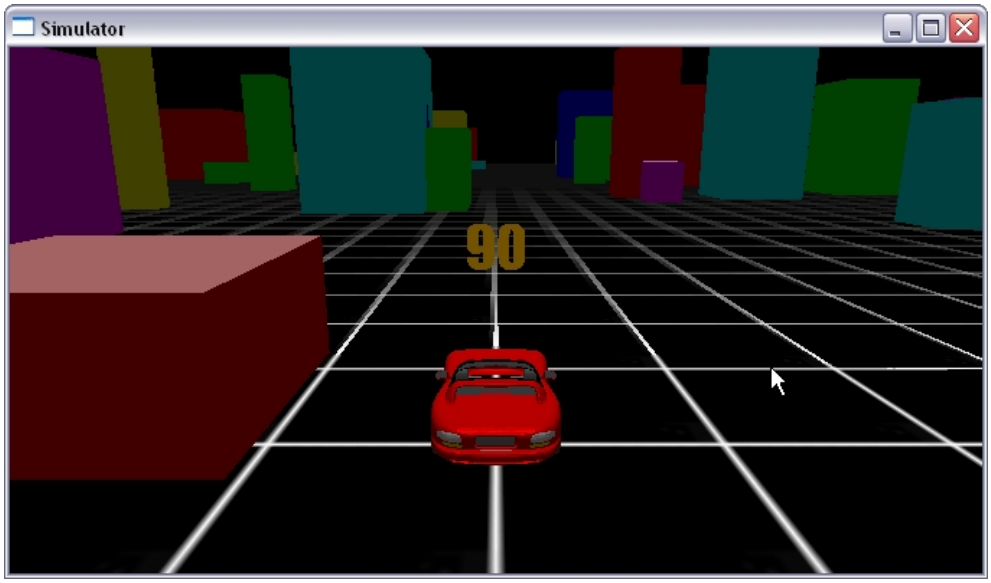
Vlastní detekce probíhá na základě znalosti umístění překážek ve scéně. V cyklu přes všechny překážky se testují všechny definované detekční body, zda se některý z nich nedostal dovnitř překážky. Pokud je některý bod uvnitř některé z překážek, je detekována kolize. Simulace si s kolizí poradí tak, že nastaví pozici modelu ve scéně na minulou pozici, na níž detekce nenastala. Zároveň nastaví aktuální rychlost *Speed* na 0. Stejně nastaví na 0 i proměnné reprezentující stav ovládacích prvků *Throtle, Break* a *Steer*.

4 Demonstrační aplikace

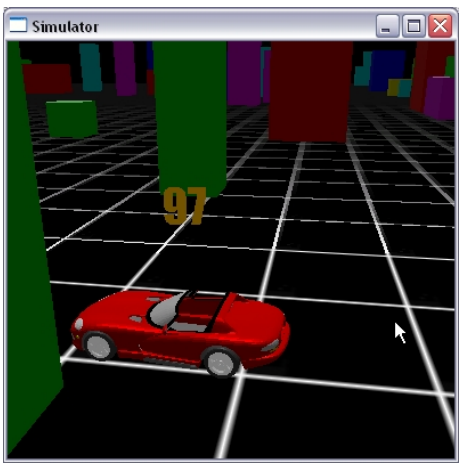
4.1 Popis aplikace

Aplikace demonstruje algoritmy popsané v kapitole 3.

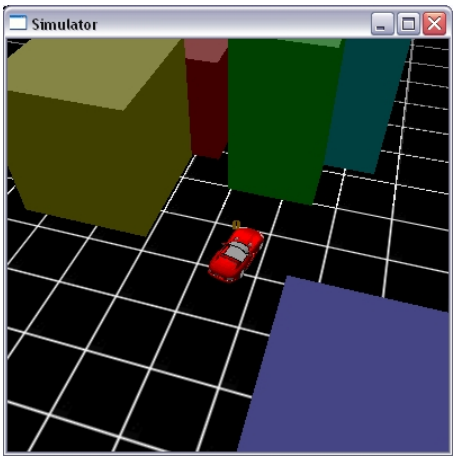
Do scény aplikace jsou náhodně generovány překážky v podobě kvádrů v 6 barvách. Ve scéně je model auta, nad nímž je zobrazena jeho aktuální rychlost. Model se dá ovládat pomocí šipek na klávesnici. Na stisk těchto kláves model reaguje tak, jak je popsáno v kapitole 3.2 . V aplikaci lze také pohybovat kamerou kolem modelu, lze ji zvedat a lze ji přibližovat k modelu. Ukázky aplikace jsou na obrázcích 4.1, 4.2 a 4.3.



obrázek 4.1



obrázek 4.2



obrázek 4.3

V aplikaci je deklarována proměnná *demage* , do které se kumuluje poškození auta způsobené kolizemi s překážkami. Čím vyšší je rychlost při kolizi, tím větší poškození

způsobí a o to více se zvýší proměnná *demage*. Pokud hodnota této proměnné přesáhne určitou mez, je model jakoby zničen a nelze s nim dále jezdit. Místo rychlosti se pak nad autem zobrazí „GAME OVER“, tak jako na obrázku 4.4. Kamerou lze pohybovat i nadále.

obrázek 4.4

4.2 Ovládání aplikace

- šipky – ovládání modelu
- numpad „2“ – pohyb kamery dolů
- numpad „8“ – pohyb kamery nahoru
- numpad „4“ – rotace kamery kolem modelu vlevo
- numpad „6“ – rotace kamery kolem modelu vpravo
- numpad „+“ – přiblížení kamery k modelu
- numpad „-“ – oddálení kamery od modelu
- numpad „5“ – nastavení pozice a rotace kamery na hodnoty jako po spuštění aplikace

5 Závěr

5.1 Zhodnocení

Navrhl jsem algoritmy pro fyzikální model a pohyb auta. S využitím těchto algoritmů jsem implementoval ukázkovou aplikaci. Z aplikace je vidět, že simulace se podobá pohybu skutečného auta. Průběh zrychlování auta z nulové rychlosti do maximální má podobný průběh jako ve skutečnosti, tedy zpočátku je přírůstek rychlosti větší, zatímco později již rychlost stoupá pomaleji. Nakonec model přestane zrychlovat úplně, protože odporová síla prostředí je již příliš velká. Méně realistické je zatáčení modelu, kdy i při velkých rychlostech je možné zatáčet o velký úhel. To je způsobeno tím, že jsem v simulaci neuvažoval odstředivou sílu. Ta by simulaci dodala na realističnosti.

5.2 Pokračování projektu

V první řadě by se dalo v projektu pokračovat vylepšením modelu auta. V modelu např. zcela chybí simulace převodovky. S převodovkou a převodovými stupni souvisí otáčky motoru, na které není v simulaci pamatováno. Pokud by se v simulaci uvažovaly otáčky motoru, měla by být také definována křivka průběhu výkonu motoru v závislosti na otáčkách.

Dalším logickým pokračováním vývoje projektu je rozšíření simulace pohybu auta i do třetího rozměru tzn. pohyb po zvlněném terénu. To by s sebou neslo potřebu přidat do všech výpočtů i výpočet třetí, tedy y-ové, souřadnice. Dále by bylo nutné při výpočtech uvažovat další síly. Byly by to síly tíhová a třecí. V souvislosti s třecí silou by bylo nutné definovat v každém bodě povrchu součinitel smykového tření. Pomocí tohoto součinitele by se zjišťovalo, zda auto např. nesklouzne ze svahu.

V návaznosti na rozšíření simulace do třetího rozměru by bylo možné do ukázkové aplikace přidat generování náhodného terénu nebo projekt přímo spojit s jiným projektem zabývajícím se generováním města (např. Virtual town generator na adrese http://merlin.fit.vutbr.cz/wiki/index.php?title=Inventor_Projects_2005).
Aplikace by se pak mohla rozšířit na hru ve stylu Need For Speed apod..

5.3 Internetová prezentace

Projekt je prezentován společně s ostatními projekty ak. roku 2005/2006, které vede Ing. Pečiva, na stránce
http://merlin.fit.vutbr.cz/wiki/index.php?title=Inventor_Projects_2006

Literatura

[1] knihovna Coin3D.
URL: http://www.coin3D.org (květen 2006)

[2] dokumentace ke knihovně Coin3D.
URL: ftp://ftp.coin3d.org/pub/coin/doc/Coin-doc.zip (květen 2006)

[3] Wernecke, Josie: The Inventor Mentor, Addison-Wesley, 1994, ISBN 0201624958

[4] Ing. Pečiva, Jan: Open Inventor tutorial.
URL: http://www.root.cz/clanky/open-inventor (květen 2006)

[5] Pochop, Jiří: Simulace pohybu auta [ročníkový projekt]. Vysoké učení technické v Brně, Fakulta informačních technologií
URL: http://merlin.fit.vutbr.cz/wiki/index.php?title=Inventor_Projects_2005 (květen 2006), projekt je zde pod názvem Car movement simulation

[6] popis licence Public Domain.
URL: http://en.wikipedia.org/wiki/Public_domain (květen 2006)

[7] prof. RNDr. Vachek, Jaroslav, CSc. a kol.: Fyzika pro I. ročník gymnázií, Státní pedagogické nakladatelství, Praha, 1984