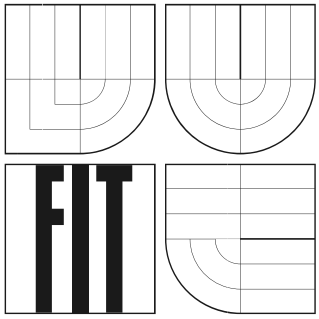
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



# Simulace pohybu auta ve virtuální scéně

Diplomová práce

2006 Jiří Pochop

# Simulace pohybu auta ve virtuální scéně



## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jana Pečivy. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Pochop
20. května

# Abstrakt

Cílem této práce je vytvořit simulaci pohybu auta po nerovném terénu ve 3D scéně za pomoci knihovny pro simulaci fyzikálních interakcí OPAL a knihovny Coin3D. Navrhnout a implementovat fyzikální model auta a vytvořit algoritmy pro jeho ovládání.

Na tomto základě potom vytvořit jednoduchý program a demonstrovat použití získaných algoritmů.

# Klíčová slova

OpenGL, Coin3D, OPAL, ODE, simulace, 3D scéna, generování krajiny, metoda zlomů, výšková mapa, transformační matice.

# Poděkování

Rád bych poděkoval svému vedoucímu, kterým byl pan Ing. Jan Pečiva, že mi při práci na projektu pomohl, že napsal tak srozumitelný tutoriál k Coin3D, za jeho trpělivost a vstřícnost.

# Abstract

The goal of this project is to create a car movement simulation over a terain in a 3D scene with physical simulation library OPAL and Coin3D library. To invent and implement the physical abstraction of a car and create algorithms to control it.

Then create a simple program and demonstrate the usage of the algorithms.



# Keywords

OpenGl, Coin3D, OPAL, ODE, simulation, 3D scene, landscape generation, fault formation algorithm, height map, transformation matrix.

# Obsah

# 1 Úvod

Tato diplomová práce si klade za cíl vytvořit funkční aplikaci využívající knihovny pro simulaci fyzikálních interakcí OPAL a knihovny Coin3D pro vytvoření simulace pohybu modelu auta ve virtuální scéně. V první řadě jde tedy o prozkoumání možnosti současného použití výše uvedených knihoven pro dosažení cíle. Knihovna Coin3D je dostatečně dokumentovaná a v ostatních projektech hojně používaná, zaměřil jsem se tedy ve své práci na vyzkoušení vlastností knihovny OPAL a její vhodnost pro využití v dalších projektech založených na zobrazování knihovnou Coin3D.

Prvním krokem provedeným v ročníkovém projektu v r. 2005 bylo seznámení se s knihovnou Coin3D, navržení algoritmů pro pohyb auta po nerovném terénu ve virtuální scéně a demonstrace zjištěných poznatků v jednoduché aplikaci. Tento projekt byl doveden do úspěšného konce a prokázal použitelnost knihovny Coin3D pro intuitivní avšak dostatečně kvalitní zobrazování 3D scén. Ovšem celý systém aplikace a interakce objektů ve scéně (terénu a auta) byl tvořen na základě matematických výpočtů a ne zcela odpovídal realitě. Za zmínku stojí také nadprůměrná spotřeba systémových prostředků na výpočty simulace.

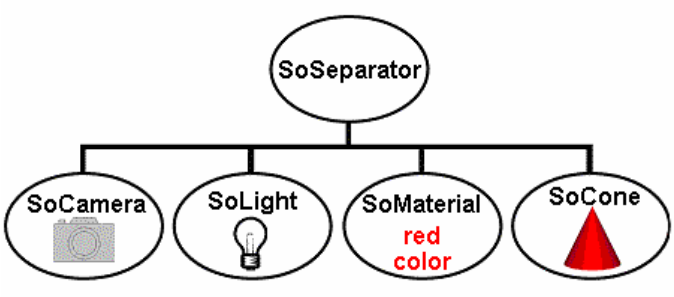
Algoritmy vyvinuté v ročníkovém projektu byly vhodné pouze pro dosazování modelů aut (po drobných úpravách i jiných objektů) do scény za účelem oživení scény pohyblivým modelem a nejednalo se o „simulaci“ v pravém slova smyslu.

Knihovna fyzikálních interakcí OPAL se tedy nabízela jako další logický krok k dosažení požadovaného efektu simulace pohybu auta po nerovném terénu. Ovšem knihovna OPAL není tak rozšířená a nebylo jasné, podaří-li se skloubit systém zobrazování Coin3D a fyzikálního simulátoru OPAL. Tuto otázku jsem rozpracoval v semestrálním projektu v r. 2006. Podařilo se mi proniknout do struktury knihovny OPAL a vytvořit jednoduchou aplikaci simulující interakce základních primitivních objektů (krychle, koule apod.).

Klíčové pro další práci se systémem OPAL byly dvě věci: že se do fyzikální simulace podařilo převést složitou geometrii nerovného terénu a nalezení způsobu přenosu informace o pohybu modelů z fyzikální simulace do systému zobrazení Coin3D. Tím se tedy potvrdila kompatibilita obou systémů a jejich použitelnost v jedné aplikaci společně.

# 2 Coin3D

Coin3D je nadstavbou grafického systému OpenGL. Jeho použitím se práce s 3D grafikou výrazně zjednodušuje. Využívá stromové struktury pro vyjádření závislostí jednotlivých komponent scény. Scéna má kořenový uzel a všechny komponenty scény (modely aut, model terénu, světelný zdroj, kamera atd.) jsou jeho synové (nodes). Do tohoto grafu scény se zařazují také „nehmotné“ prvky scény: kamera, světla, časovač atp. Vhodným skládáním jednotlivých jednoduchých komponent může tedy jednoduše vzniknout komplexní simulace na vysoké úrovni.




*Obrázek 1. Příklad grafu scény. Převzato z [1]*

## 2.1 Struktura aplikace postavené na Coin3D

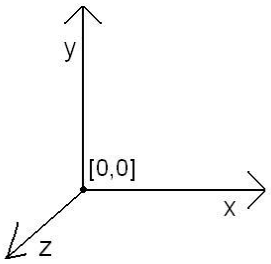
Aplikace vytvořená na základě knihovny Coin3D má svoji specifickou strukturu. Prvním krokem je inicializace Coin3D. Bez tohoto kroku nelze přidávat do scény žádné objekty a aplikace nemá ještě přiřazeno žádné okno, do kterého bude renderovat. Druhým krokem je sestavení stromu objektů a nastavení světel a kamer. Coin3D zde nabízí takřka vyčerpávající možnosti nastavení jednotlivých komponent z hlediska OpenGL. Ve třetím kroku se zobrazí okno aplikace, rozjede se nekonečná renderovací smyčka a té se předá právě vytvořený strom objektů.

Tato hlavní smyčka je už interní součástí knihovny Coin3D a programátor by tedy neměl šanci ovlivnit další průběh zobrazení scény. K tomu, aby bylo možno například reagovat na podněty z klávesnice nebo myši, je hlavní renderovací smyčka COIN vytvořena jako událostmi řízená aplikace – je možno do stromu scény vložit uzel časovač, který v naplánovaný okamžik předá řízení funkci, která je mu v první fázi tvoření programu zaregistrována. V této funkci lze ošetřit všechny případné vstupy a v návaznosti na ně potom upravit běh aplikace (např.: ukončit aplikaci, přidat objekt do scény, upravit transformační matici některého z objektů apod.).

Ke zpracování vstupů z klávesnice slouží speciální uzel, který v průběhu renderování zpracovává vstupy z klávesnice a tyto lze ve funkci registrované časovačem ošetřit a příslušně zareagovat.

## 2.2 Scéna Coin3D

Scénu tvoří „nekonečný“ prostor, do kterého jsou vkládány jednotlivé objekty scény. Jejich poloha je definována třemi souřadnicemi pravoúhlého souřadného systému.




*Obrázek 2. Souřadný systém scény Coin3D.*

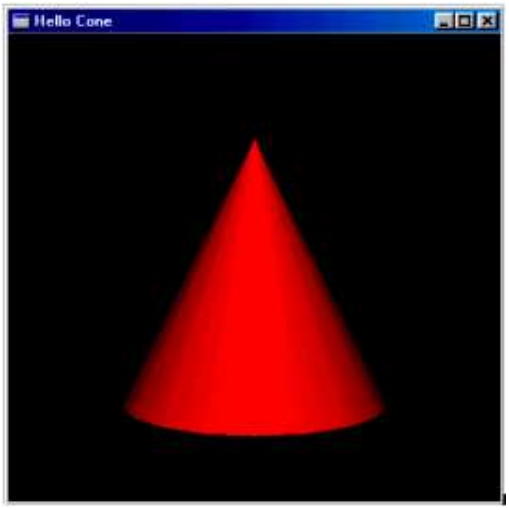
## 2.3 Příklad jednoduché aplikace s Coin3D

Zde bych shrnul některé věci z této kapitoly a pokusil se nastínit skutečnou strukturu aplikace založené na knihovně Coin3D:

```
Okno := Inicializuj_Coin3D();
Root := Vytvoř_kořenový_uzel_scény();

Kamera := Vytvoř_kameru(parametry_kamery);
Root -> Vlož_uzel_jako_svého_syna(Kamera);
Světlo := Vytvoř_světlo(parametry_světla);
Root -> Vlož_uzel_jako_svého_syna(Světlo);
Materiál := Vytvoř_materiál(parametry_materiálu);
Root -> Vlož_uzel_jako_svého_syna(Materiál);
Kužel := Vytvoř_kužel(parametry_kužele);
Root -> Vlož_uzel_jako_svého_syna(Kužell);

Zobraz_okno_a_rozjeď_renderovací_smyčku(Okno, Root);
```




Tato aplikace by zobrazila jednoduché okno a v něm by bylo možno vidět kužel, který by měl barvu definovanou uzlem „Materiál“, na nějž by svítilo světlo podle nastavení uzlu „Světlo“ a směr a pozice, ze které bychom kužel sledovali by byly definovány nastavením uzlu „Kamera“. Výsledek by se mohl podobat tomu zachycenému na přiloženém obrázku.

Tento postup (i když byl značně zjednodušen) a obrázek byly přejaty z [1].

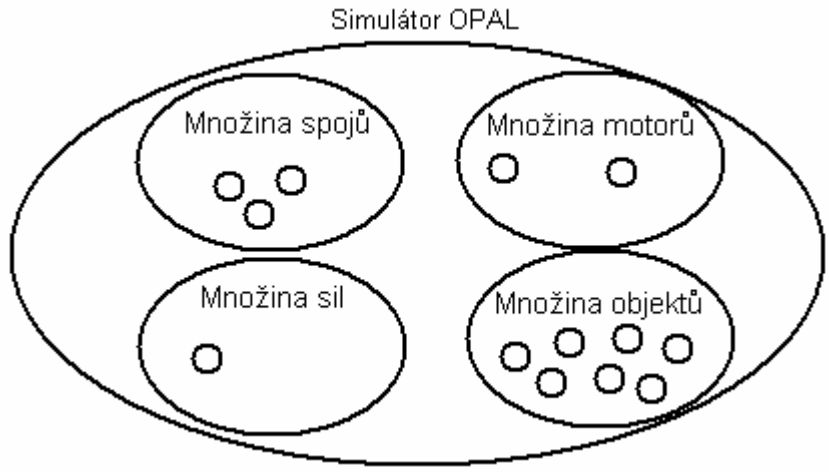
# 3 OPAL

Knihovna OPAL (Open Physics Abstraction Layer) je nadstavbou nad systémem fyzikálních knihoven ODE [5]. Za pomoci této knihovny je možné vytvářet vztahy mezi objekty zahrnutými v simulaci a efektivně v reálném čase simulovat jejich vzájemné působení (přitažlivé síly, kolize atd.).

Na rozdíl od systému Coin3D není součástí simulátoru nekonečná smyčka, ale pouze metoda simulátoru, která provede simulaci na předem zvoleném časovém intervalu - simulace probíhá po jednotlivých krocích.

## 3.1 Prostor simulace Opal

Fyzikální simulace v knihovně Opal využívá pro definici objektů a jejich vzájemných vztahů stromovou strukturu uzlů různých typů a funkcí. Základem celé simulace je instance třídy *Simulator*, do které se postupně vkládají další objekty. Tyto mohou být různých druhů: do simulace můžeme vkládat **motors** (motory) – jsou původci pohybu předmětů ve scéně, **joints** (spoje) - působí jako vazby mezi objekty, **solids** (pevné látky) – představují objekty simulované simulátorem.




*Obrázek 3. Struktura simulátoru OPAL*

### 3.1.1 **Motors** (motory)

Motorů je v knihovně Opal k dispozici několik. Liší se svými vlastnostmi a účinky na objekty. Nalezneme zde například abstrakci **raketového motoru** (působí na dané těleso konstantní silou daným směrem), **servo motoru** (působí pouze v jedné ose, umožňuje přesné natočení), **motoru auta** (simuluje motor auta – parametry jako jsou maximální rychlost a maximální otáčky lze definovat chování motoru. Při maximální rychlosti má minimální otáčky).

Motory se připojují k jednotlivým osám spojů a aplikují na ně svůj účinek. Nabízí dostatečný počet parametrů k vytvoření široké škály různých druhů motorů.

### 3.1.2 **Joints** (spoje)

Omezují stupeň volnosti připojených objektů: objekty mají 6 stupňů volnosti (3 lineární a 3 rotační). V závislosti na svém druhu omezuje připojené objekty v některých stupních volnosti (např. způsobí, že se objekty mohou vůči sobě jenom otáčet kolem jedné osy). Další jejich vlastností je, že tyto vazby mohou být poškozeny a v důsledku toho se vazby poruší (toto záleží na jednotlivých nastaveních a je možné vytvořit nezničitelné vazby).

Každý spoj má definovány tři osy (tři směry, podle kterých se omezuje pohyb připojených objektů) a tzv. anchor-point (bod uchycení, ukotvení), který definuje bod, ve kterém spoj aplikuje svá omezení. Každý z několika různých druhů spojů používá jinou kombinaci těchto čtyř definovaných vlastností pro vytvoření požadovaného efektu.

- **Hinge joint** (otočný spoj) – využívá první osu a bod ukotvení k simulaci spojení, které připojeným objektům povolí pouze otáčení podle definované osy.
- **Universal joint** (univerzální spoj) – využívá dvě osy a bod ukotvení
- **Ball joint** (kloubový spoj) – využívá všech tří os a bodu ukotvení k simulaci úplného kloubového spoje.
- **Slider joint** (posuvný spoj) – využívá pouze jedné osy k simulaci spoje, který povolí svým objektům se vůči sobě pouze posunovat.
- **Wheel joint** (spoj pro kolo) – využívá dvě osy – jednu pro točivý pohyb a druhou pro řídící osu a používá také bod ukotvení.
- **Fixed joint** (pevný spoj) – nevyužívá žádnou definovanou osu a ani bod ukotvení a nepovoluje připojeným objektům žádný vzájemný pohyb.

### 3.1.3 **Shapes** (tvary)

Jsou nositeli informace o tvaru objektu, obsahují geometrii objektu. Nemohou existovat samostatně, vždy jsou součástí objektu „solid“, definují jeho tvar a představují kolizní těleso pro simulaci. Těchto tvarů může být do jednoho objektu vloženo více, pro definování složitějšího tělesa. V závislosti na typu tvaru má každý definován potřebné parametry (pozice uvnitř objektu, rozměry a materiál) zároveň s odpovídající povrchovou reprezentaci (síť trojúhelníků podobná struktuře pro renderování). Zde můžeme využít několik málo předdefinovaných tvarů (krychle, koule apod.), nebo zadat svoji reprezentaci složitějšího objektu ručně, pomocí ruční definice jednotlivých vertexů a trojúhelníků:

**Pole vertexů:** každé tři po sobě jdoucí buňky obsahují informaci o jednom bodě (jeho souřadnice x, y, z)

**Pole trojúhelníků:** každé tři po sobě jdoucí buňky definují jeden trojúhelník (indexy do pole vertexů určující tři body definující trojúhelník)

**Počet vertexů:** počet bodů, *velikost_pole_vertexů = počet_bodů x 3*

**Počet trojúhelníků:** *velikost_pole_trojúhelníků = počet_trojúhelníků x 3*

### 3.1.4 **Solids** (objekty)

Každý „solid“ reprezentuje jeden objekt v prostoru simulace. Obsahují informace o pozici objektu ve scéně a jeho ostatní fyzikální parametry (např. jestli se má objekt zahrnout do detekce kolizí, nebo ne). Do této „obálky“ se vkládají jednotlivé tvary, které definují vlastní tvar objektu a detaily povrchu pro detekci kolizí mezi objekty. Těchto tvarů se do objektu může vložit více, například pro definici složitějšího objektu.

Důležitou vlastností objektu je možnost označit objekt jako statický, což při vlastní simulaci zrychluje jeho zpracování. Zvláště vhodné je to například u větších nebo složitějších předmětů u kterých víme, že se nebudou pohybovat.

Velice zajímavá se ukázala možnost rozdělit objekty do různých skupin a nastavit, které skupiny mezi sebou budou kolidovat a které ne.

### 3.1.5 **Forces** (síly)

Definují síly působící na tělesa. Mimo gravitace můžeme nechat na těleso působit jakoukoli sílu, kterou definujeme. Mohou to být síly lokální nebo globální, působící neustále, nebo jen po určitou dobu. Také můžeme definovat přesné působiště síly uvnitř objektu, nebo nechat sílu působit na těleso v jeho těžišti (globálně).

### 3.1.6 **Materials** (materiály)

Každý „shape“ má definovány určité vlastnosti, které ovlivňují jeho chování ve scéně, zvláště pak když dojde ke kolizi dvou objektů. Pomocí materiálů definujeme

**hustotu** - v závislosti na rozměrech a hustotě se určí váha tělesa

**tvrdost** - určuje, do jaké míry se předměty mohou deformovat (prolínat) při kolizi

**elasticitu -** podle tohoto parametru se při srážce odvodí velikost síly odrazu

**klouzavost** – určuje, jaký odpor klade povrch při tření o jiný objekt

Tyto parametry nabývají hodnot mezi 0 a 1 a jejich kombinace umožňují simulovat objekty vytvořené z různých materiálů. Je velice důležité pro správný průběh simulace nastavit správné parametry materiálů.

## 3.2 Struktura aplikace s OPAL

Knihovna OPAL obsahuje pouze nástroje k simulaci dějů mezi zadanými objekty a pro jejich zobrazení je zapotřebí jiného nástroje. Lze ovšem sestavit takovou aplikaci, která obsahuje pouze simulaci.

Aplikace využívající simulátor OPAL má také svoji specifickou strukturu. Jako první se musí vytvořit instance simulátoru, který se musí inicializovat a nastavit tak některá globální nastavení

simulace. Mezi tyto patří nastavení jedné globální síly, která v simulaci zastupuje gravitační sílu, nastavení přesnosti simulace (máme na výběr ze pěti úrovní) a nastavení velikosti jednoho kroku simulace.

Přidávání jednotlivých objektů do simulace se provádí metodami třídy Simulator, která představuje hlavní entitu v simulaci. Voláním těchto funkcí se nejen vytvoří instance požadovaného objektu, ale objekt se také zaregistruje do simulace. Objektu se po vytvoření musí přiřadit transformační matice a vložit do něj jednu nebo více datových struktur popisujících tvar objektu.

V dalším kroku definujeme vazby mezi objekty, síly které na objekty působí, motory a jejich vlastnosti a máme možnost registrovat si několik druhů callbacků pro případnou obsluhu vybraných událostí.

Zbývá už jen rozjet simulaci a to vytvořením smyčky, ve které voláním metody simulátoru *simulate* provádíme jednotlivé kroky simulace. Po ukončení simulace je zapotřebí odregistrovat všechna data (odstranit je tím z paměti) a zničit simulátor.

## 3.3 Příklad jednoduché aplikace s OPAL

Nyní bych se pokusil názorně shrnout věci popsané v této kapitole nastíněním skutečné struktury aplikace používající OPAL.

```
Simulátor: = Vytvoř_simulátor_OPAL();
Simulátor -> init(gravitace, krok_simulace, přesnost_simulace);
Solid := Simulátor -> Vytvoř_solid();
Transformace := Vytvoř_matici_transformace();
Solid -> Nastav_matici_transformace(Transformace);
Shape := Vytvoř_shape_krychle();
Shape -> Nastav_parametry(rozměry, materiál, pozice);
Solid -> Vlož_shape(Shape);

Opakuj_nekonečně:
        dt = uplynulý_čas_od_minulého_volání();
        Simulator -> simulate(dt);
```

Tento program by však nereagoval na vnější vstupy a ani by nevykresloval aktuální děje. Ošetření vstupů a vykreslování by se muselo zařídit jinými prostředky uvnitř nekonečné smyčky.

# 4 Postup prací na projektu

## 4.1 Vytvoření nerovného povrchu

Nerovný povrch využitý v demonstrační aplikaci je vytvořen algoritmy převzatými z článku „*Seriál Open Inventor*“ [1] a pro účely demonstrace vyvinutých algoritmů plně vyhovuje, naopak je přímo ideální – terén se generuje při každém spuštění aplikace znova a minimalizuje se tedy možnost chyby testováním pouze na jednom zakřiveném povrchu.

Volba tohoto algoritmu se ukázala jako velice výhodná nejen v průběhu ročníkového projektu, ale i v navazujících projektech – v semestrálním projektu a nakonec i v diplomové práci. Datové struktury (i když mírně upravené a doplněné) poskytly základ pro matematické výpočty polohy modelu auta v ročníkovém projektu a později prokázaly svoji výbornou kompatibilitu s datovými strukturami pro převod geometrie do simulátoru OPAL. Více o této konverzi v dalších kapitolách

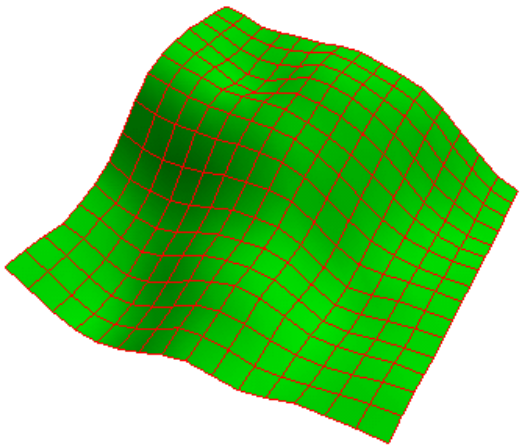
### 4.1.1 Algoritmus

Pro modelování terénu pro tento projekt jsem využil Fault formation algoritmus (metoda zlomů). Používá velmi jednoduchý princip - náhodně vygeneruje "zlom" v krajině a pak ke všem bodům jedné poloviny připočte náhodnou hodnotu ([1] – článek: „Terrain Generator 2“). Nakonec je celý povrch vyfiltrován filtrovací funkcí, aby bylo dosaženo hladšího a přirozenějšího vzhledu.

### 4.1.2 Datová struktura

Vygenerovaná datová struktura obsahuje model krajiny se všemi daty o trojúhelnících a normálách, ale pro potřeby dalšího využití tohoto modelu jako podkladu pro scénu s pohybujícími se auty především obsahuje tzv. **výškovou mapu.** Bylo by neekonomické z hlediska paměťové náročnosti uchovávat informace o výšce každého bodu ve scéně, proto je výšková mapa pouze dvourozměrné pole hodnot, které uchovává informace o výšce bodů ve vrcholech základní mřížky modelu krajiny.

Mezi těmito základními body výšku aproximujeme z okolních bodů.



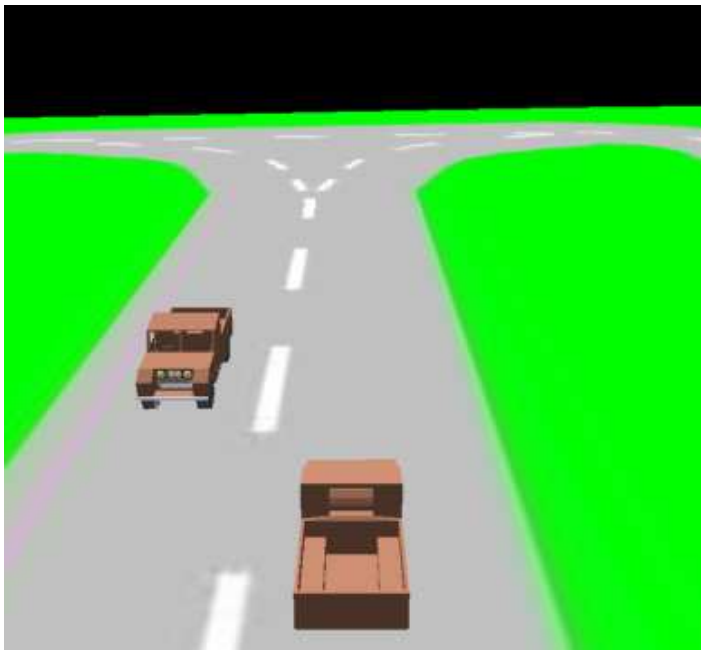
*Obrázek 4. Výšková mapa terénu. Převzato z [1]*

## 4.2 Použití nerovného terénu s Coin3D

Stejně jako generování nerovného povrchu, tak i jeho zařazení do grafu scény Coin3D jsem převzal z článku *„Seriál Open Inventor“* [1]. Pro použití se systémem Coin3d v aplikaci pro simulaci pohybu auta nebylo potřeba zde navržené algoritmy nijak upravovat.

Velice elegantním postupem je celý nerovný terén vložen do grafu scény Coin3D a je v každém kroku zobrazován, jako součást scény.

Ročníkový projekt měl za cíl matematicky dosadit model do scény, tedy se již dále zpracováním modelu terénu nezabýval. Důležitá z hlediska navazujících projektů zde byla implementace některých funkcí a datových struktur zjednodušujících práci s výškovou mapou. Také aplikace textur na různé povrchy v Coin3D jsem si v tomto projektu měl možnost vyzkoušet s výhledem na zúročení této zkušenosti v budoucnu.



*Obrázek 5. Výsledná aplikace ročníkového projektu*

Ročníkový projekt se sice ubíral jiným směrem, ale některými rozhodnutími, která padla v průběhu jeho řešení, výrazně ovlivnil a usnadnil vývoj navazujících projektů.

## 4.3 Algoritmus simulace vyvinutý v RP

V této kapitolce bych rád uvedl zjednodušenou blokovou strukturu algoritmu vyvinutého jako hlavní smyčka programu založeného na knihovně Coin3D – jedná se o funkci, která se vykonává v rámci nekonečné smyčky renderování sytému Coin3D (viz. Kapitola 2) jako reakce na událost časovače vloženého do grafu scény:

```
Při_spuštění_callbacku_časovače:
    dt:=zjisti_uplynulý_čas();
    Ošetři_vstupy_z_klávesnice();
    Proveď_výpočty_na_modelech(dt);
    Uprav_objekty_scény_podle_nových_parametrů();
    Vykresli_scénu();
```

*Zjednodušený algoritmus enginu simulace*

## 4.4 Spolupráce OPAL a Coin3D

Systém Coin3D se pro zobrazování 3D scény dokonale osvědčil, zbývalo pouze vybrat vhodnější „funkční“ část simulace. Výběr padnul na zatím ne moc známou knihovnu fyzikálních interakcí OPAL. Po bližším prozkoumání struktury knihovny OPAL (viz. kapitola 3) se ukázalo, že knihovna OPAL by mohla být s knihovnou Coin3D kompatibilní – knihovna OPAL jako prvek simulující vzájemné působení objektů ve scéně a systém Coin3D pro jejich zobrazování.

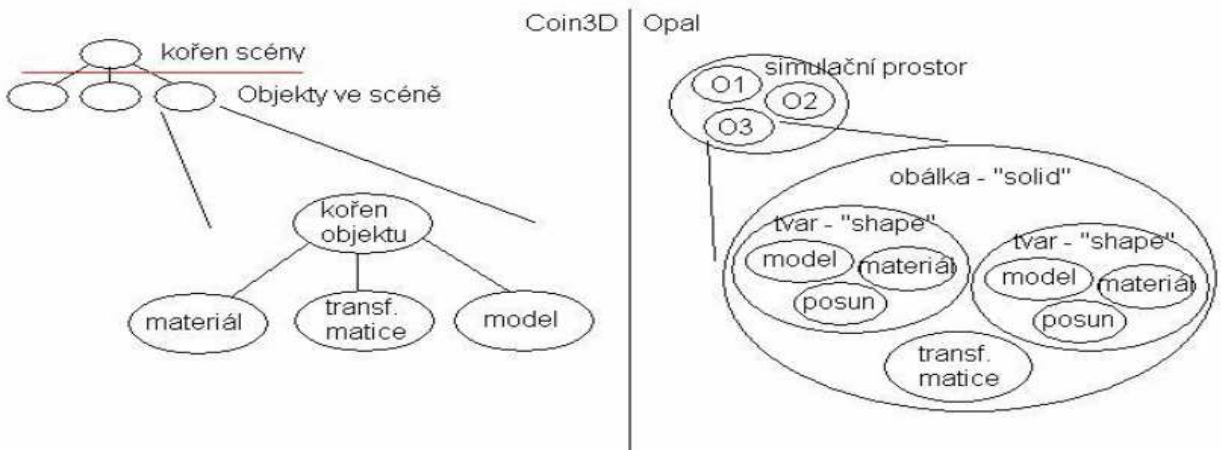
### 4.4.1 Převod geometrie do systému OPAL

Objekty simulované systémem OPAL mohou mít tvary předdefinované knihovnou OPAL (jsou to pouze jednoduché tvary: koule, kvádr apod.) nebo mohou mít tvar definovaný uživatelem. Pro simulaci modelu auta postačí zmíněné jednoduché objekty jako abstrakce jednotlivých tvarů – kolizní těleso v simulaci není vidět a stačí, aby tvarově zhruba odpovídalo tvaru modelu. Dají se tedy například místo modelů kol použít koule.

Složitější objekty, například kolizní model nerovného terénu, je možno do systému OPAL zadat definicí trojúhelníkové sítě (viz. kapitola 3). V tomto kroku bylo nutné zasáhnout do algoritmu generování krajiny (viz. kapitola 4.1) a do pomocných datových struktur (podle definice OPAL) uložit potřebná data o počtech trojúhelníků, souřadnice jednotlivých bodů apod. Zde se opět prokázala výhodnost použitého algoritmu generování krajiny, protože postup generování krajiny touto metodou přesně koresponduje se specifikacemi uživatelského objektu v OPAL. Nebylo tedy zapotřebí žádné větší a složitější konverze.

### 4.4.2 Převod informací o poloze objektů do Coin3D

V každém simulačním kroku se na základě vzájemných interakcí objektů a parametrů simulátoru vypočítává nová poloha objektů. Jak tedy přenést informace z odsimulovaného prostředí do systému pro zobrazování? Odpověď nalezneme při bližším pohledu na datové struktury obou systémů.




*Obrázek 6. Datové struktury systémů Coin3D a OPAL*

Jak je z výše uvedeného obrázku (6) patrné, každému objektu ve scéně náleží model (nebo skupina modelů v případě OPAL, což ale nyní není tak důležité, definují jeden objekt), materiál (v případě materiálu Coinu se jedná o světelné vlastnosti, v případě OPALu o vlastnosti fyzikální) a transformační matice [3], která určuje pozici, měřítko a natočení modelu ve všech třech osách. A právě přes transformační matici lze převést údaje o modelu ze systému OPAL do systému Coin3D. Pro zobrazení simulovaných objektů tedy stačí po každém kroku simulace převést všechny transformační matice simulovaných objektů objektům zobrazovaným.

Velice důležitým aspektem je kompatibilita obou souřadných systémů. Simulátor OPAL používá stejný souřadný systém, jako knihovna Coin.

### 4.4.3 Hlavní smyčka programu

Jako výhodné se ukázalo použití hlavní renderovací smyčky Coin3D vyvinuté v rámci RP a drobně upravené tak, že místo výpočtů pozic modelů se provede simulace na jejich protějších v systému OPAL a následně se převedou informace z transformačních matic do objektů Coin3D.

```
Při_spuštění_callbacku_časovače:
    dt:=zjisti_uplynulý_čas();
    Ošetři_vstupy_od_uživatele();
    Uprav_parametry_simulovaných_objektů();
    Proveď_simulaci_Opal(dt);
    Převeď_transformace_objektů_do_stromu_coin3d();
    Vykresli_scénu();
```

*Zjednodušený algoritmus enginu simulace*

### 4.4.4 Přínos semestrálního projektu

Kompatibilita knihoven Coin3D a OPAL při vzájemné spolupráci při simulaci a zobrazení 3D scény se ukázala být téměř dokonalá. Přenos geometrie do simulátoru, přenos informací o pohybujících se objektech zpět do zobrazovacího systému je tímto vyřešen a zbývá prozkoumat možnosti knihovny OPAL. Na přiloženém obrázku (7) vidíme implementaci zatím dosažených výsledků: přenesená geometrie zvlněného terénu, zobrazen je model auta a zobrazeny jsou také kvádry, představující kolizní model auta. Model auta leží na povrchu zvlněného terénu, to vše odsimulované OPALem a zobrazené grafickým systémem.

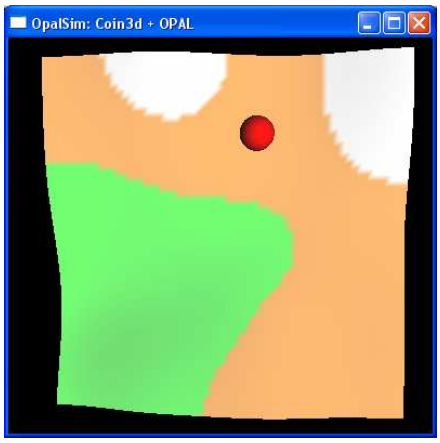
*Obrázek 7. Model auta na zvlněném terénu*

## 4.5 Jednoduchý pohyblivý model v OPAL

Doposud vytvořené aplikace měly charakter simulace volného pádu a následné kolize několika předmětů, případně s dopadem na zvlněný terén. Po překonání problému zobrazení simulovaných objektů lze přistoupit k vlastnímu modelování složitějších scén. První složitější simulací, která obsahuje pohyblivý (po terénu jezdící) předmět a reaguje i na podněty z klávesnice, byla koule, pohybující se po zvlněném terénu.

Byl to první pokus s knihovnou OPAL a nepadajícím předmětem a měl rozhodnout o použitelnosti knihovny OPAL pro další rozvoj aplikace s cílem vytvořit simulaci pohybu auta.

Do scény tedy vložíme model nerovného terénu a model koule a jejich odpovídající reprezentace v simulátoru, nastavíme požadované reakce do funkce obsluhující události z klávesnice a spustíme simulaci. Na kouli působí gravitační síla nastavená při inicializaci simulátoru a při stisku klávesy na ni začne působit síla ve směru stisknuté šipky. Tímto způsobem je možné koulí po povrchu pohybovat a nebo na ni nechat působit jen gravitační sílu.




*Obrázek 8. Aplikace s pohyblivou koulí*

Jednoduchý příklad tedy ukázal použitelnost knihovny OPAL v jednoduché interaktivní aplikaci, implikující možnost vývoje složitější aplikace.

## 4.6 Spojování objektů v OPAL

K vytvoření složitějšího modelu potřebného pro vytvoření simulace pohybu auta je zapotřebí spojit více objektů tak, aby tvarem připomínaly auto a mohly fungovat jako jeho kolizní model. První pokusy se spojováním objektů ovšem proběhly pouze na dvou koulích, které jsem se pokusil spojit různými druhy spojů, které OPAL nabízí.


*Obrázek 9. Dvě spojené koule*

Ale spojení dvou objektů, byť tak jednoduchých, se ukázalo býti velice obtížné, zvláště díky malé dokumentovanosti projektu OPAL. Po desítkách neúspěšných pokusů se ale podařilo dvě koule spojit stabilním spojem, který je ale podmíněn nastavením přesnosti simulace na maximální hodnotu (SOLVER_ACURACY_VERY_HIGH). Všechny kombinace nastavení spoje selhávají při jiném nastavení

přesnosti simulátoru. Jedná se zřejmě o vnitřní chybu simulátoru, která bude snad (podle fóra na stránkách projektu [4]) v dalších verzích opravena.

Podařilo se mi tedy vytvořit aplikaci na plochém terénu (aby byly zřetelné účinky spoje a působící síly) se dvěma spojenými koulemi. V závislosti na stisknuté šipce je na jednu z koulí aplikována síla a díky spoji mezi koulemi se dají obě do pohybu. Pro větší přehlednost a názornost byly obě koule opatřeny texturou.

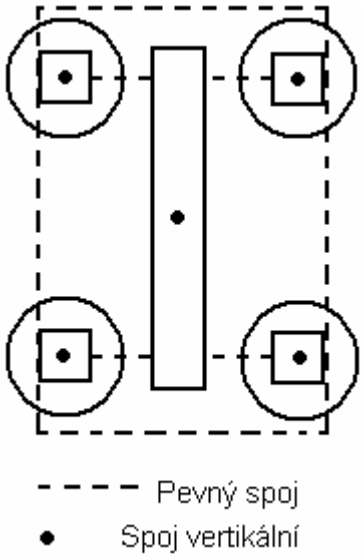
## 4.7 Složitější pospojovaný objekt v OPAL

Po vyřešení, v jednu dobu téměř fatálního, problému spojování objektů jsem se pokusil vytvořit z těchto jednoduchých objektů abstrakci modelu auta. Místo kol čtyři koule, jako kolizní těleso karosérie kvádr. U tohoto modelu už byly použity jiné druhy spojů, než „pevný spoj“. Pro připojení kol jsem použil „spoj pro kola“. Pevné spoje zůstaly u spojení částí karoserie a podvozku.

Model již má požadovaný tvar, osy kol však ještě ovládány přímo nejsou – na celý objekt je aplikována síla ve směru stisknutých šipek. Osy zadních kol jsou nezatáčivé (zablokovány omezením zatáčecí osy), přední kola mají otáčivé osy mírně omezeny. Model je tedy připraven pro konečnou fázi: vložení přímých ovládacích prvků – motorů.

Jako doplněk jsem simulaci doplnil o stěny bránící modelu vyjet z vygenerovaného terénu. Tyto stěny jsou implementovány jako další z „primitivních“ tvarů knihovny OPAL – jedná se o rovnicí definovanou rovinu v prostoru. Jako tvar může být vložena pouze do statických objektů (krajina je definována jako statická, takže to zde jde).


*Obrázek 10. Model auta tlačený sílou*

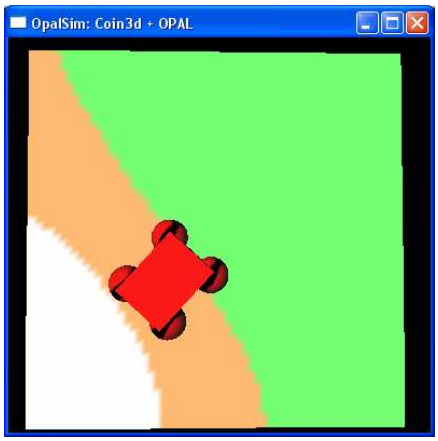



*Obrázek 11. Schéma modelu auta*

Na obrázku 11 je vyobrazeno schéma sestavení modelu auta z jednotlivých komponent. Koule symbolizující kola jsou spojem pro kola připojeny k menším krychlím uvnitř sebe. Ty jsou pevným spojem připojeny k podvozku a podvozek je pevným spojem připojen k horní části karosérie.

## 4.8 Model auta s motory

Motory, stejně jako ostatní komponenty jsou inicializovány datovou strukturou, popisující jejich vlastnosti. Přiřazují se k jednotlivým osám spojů a ovlivňují jejich pohyb. Pro pohon modelu auta je ideální použít předpřipravenou abstrakci motoru automobilového. Vytvoříme model auta s pohonem na všechny čtyři kola. Pro ovládání zatáčecích os předních kol využijeme dvou tzv. servo-motorů, které se ovládají pouze nastavením požadované pozice. Můžeme je přirovnat ke krokovým motorům.




*Obrázek 12. Klouzající model s motory*

Na chování modelu je vidět, že přesto, že je model již kompletní (žádné další komponenty už do modelu auta nepřidáme), není toto výsledek, který bychom mohli chtít. Model po ploše klouže, kola se natáčejí pomalu, nemá potřebnou přilnavost. Je to první verze modelu auta připravená pro ladění parametrů materiálů a nastavení jednotlivých motorů.

## 4.9 Vyladěný model auta

Správnou kombinaci jednotlivých parametrů všech komponent modelu se nakonec podařilo nalézt a model auta se po nerovném povrchu pohybuje jistě a neklouže jako v předcházející verzi. Svou úlohu tu hraje i nastavení správných materiálů kol a terénu a detailnější nastavení parametrů motorů.

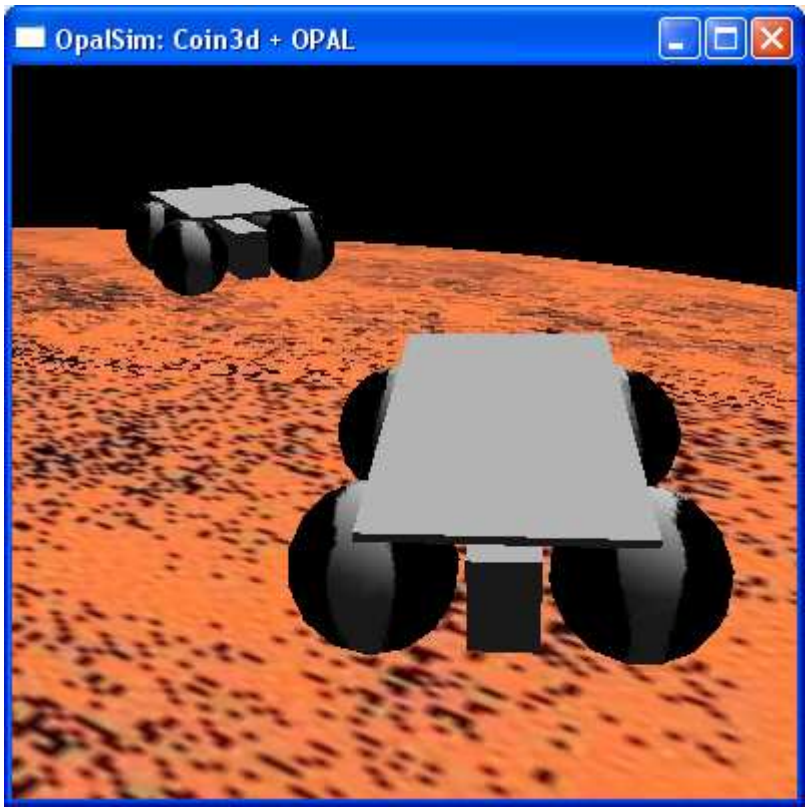
U motorů je zvláště důležité najít správné nastavení točivého momentu vzhledem k váze celého modelu. Toto je ztíženo také tím, že okamžitý točivý moment závisí také na nastavení maximální rychlosti motoru. U servomotorů ovládajících zatáčení je vhodné nastavit točivý moment co nejvyšší, aby se kola natáčela i když je na ně aplikována větší tíha.

Nastavení materiálů je potřeba věnovat také velkou pozornost – mohou ovlivnit simulaci nejen klouzáním modelu po ploše, ale také například jeho přílišným poskakováním.

Nastavení těchto parametrů je specifické pro různé velikosti a druhy modelů. Je potřeba experimentálně zjistit, jaké hodnoty parametrů jsou v dané situaci nejvhodnější. Několik poznámek zjištěných experimenty:

- Všechny parametry materiálů nabývají hodnot mezi 0 a 1. Použití mezních hodnot by mělo simulaci urychlit
- Parametr motoru auta „throttle“ musí také nabývat pouze hodnot mezi 0 a 1.
- Parametr simulátoru určující maximální počet kontaktů by měl být nastaven na hodnotu 100 a vyšší.

- Nelze vytvořit stabilní model obsahující spoje, když je přesnost simulátoru nastavena jinak, než na „SOLVER_ACURACY_VERY_HIGH“.
- Model auta po terénu tvořeném sítí trojúhelníků nejede plynule, pokud jsou trojúhelníky větších rozměrů (relativně k velikosti kol)




*Obrázek 13. Kolizní modely aut na nerovném terénu*

# 5 Ukázková aplikace

## 5.1 Datové struktury v ukázkové aplikaci

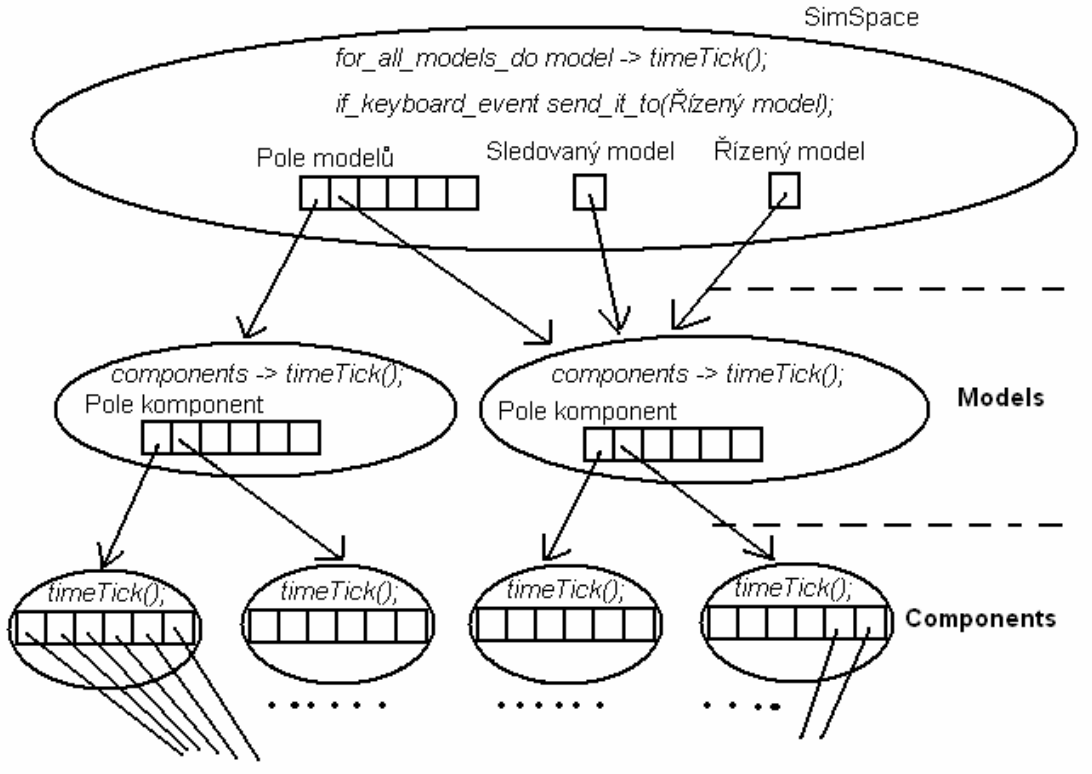
Součástí tohoto diplomového projektu je i navržení optimálního způsobu práce se simulovanými a zobrazovanými objekty. Ze závěrů kapitol 2 a 3 lze odvodit, že složitost aplikace se s přibývajícím počtem objektů neudržitelně zvětšuje. Programátor díky tomu, jak jsou oba dva systémy (OPAL i Coin3D) navrženy, velice rychle ztrácí přehled o všech objektech a žádný ze systému také přímo neobsahuje správu modelů složených z více jednoduchých objektů.

Navrhnul jsem tedy datové struktury (třídy), které by měly programátorovi pomoci se zorientovat a jednoduše vytvářet a spravovat větší množství složitějších modelů.

- **Třída Component** uchovává informace o jednom jednoduchém modelu. Udržuje ukazatele na všechny uzly grafu scény Coin3D a všechny uzly OPAl tohoto modelu. Jsou to především transformační matice pro oba systémy, potom pro Coin3D: materiál, model a kořenový uzel, pro OPAL ukazatel na vlastní objekt a na tvar. Dále poskytuje

rozhraní pro nastavení těchto atributů, důležitá je metoda nastavení pozice komponenty a metoda „timeTick“, ve které komponenta převede vlastní transformační matici OPALu do transformační matice Coinu3D.

- **Třída Model** udržuje informace o jednom celém modelu v rámci scény. Každý model se skládá z několika komponent. Ukazatele na ně jsou uloženy ve zvláštním poli, do kterého se každá komponenta musí zaregistrovat metodou „ComponentPushBack“. Obsahuje také ukazatele na všechny řídící a hnací motory modelu. Obsahuje ukazatele na komponenty, které představují pozici kamery a místo, které kamera sleduje. Zvláštní metodou je metoda „ComponentsTimeTick“, která zaručí, že všechny zaregistrované komponenty provedou své „timeTick“ metody. Na úrovni této třídy se také řeší ovládání motorů – rozhraní této funkce nabízí několik obslužných metod pro různé události od uživatele.
- **Třída SimSpace** v sobě zapouzdřuje ukazatele na objekty nejvyšší úrovně – na simulátor OPALu a na kořenový uzel scény Coin, uzel globální kamery scény a světla ve scéně. Obsahuje pole ukazatelů na všechny instance typu model a zajišťuje provedení jejich metod „timeTick“ v metodě obsluhující události časovače. Uchovává ukazatel na model, který je právě sledován kamerou a ukazatel na model, který má být právě ovládán klávesnicí. Podle tohoto ukazatele rozděluje příchozí události uzlu obsluhujícího události klávesnice. Toto všechno se děje za pomoci metody, která má strukturu podobnou té, která byla navržena v kapitole 4.4.3.




Ukazatelé na uzly do stromu Coin a na objekty simulátoru OPAL

## 5.2 Struktura ukázkové aplikace

Ukázková aplikace využívá všech navržených datových struktur k vytvoření simulace pohybu auta po nerovném terénu. V hlavním souboru aplikace, „opalsim.cpp“, je definovaná funkce „addCarModel“. Uvnitř této funkce se lineárně vytváří výše popsaný model auta podle několika zadaných parametrů tak, aby vkládání modelu do scény proběhlo pro programátora co možná nejkomfortněji a aby se vložení dalo opakovat.

V hlavním programu se vytvoří a inicializuje instance třídy „SimSpace“, která představuje simulační prostor, do kterého se budou vkládat modely. Dalším krokem je vytvoření modelu nerovného terénu. Toto nebylo přesunuto do funkce, protože se předpokládá, že pro demonstrační účely bude stačit pouze jeden model terénu. Nerovný terén je doplněn blokujícími stěnami na okrajích a jedním skokánkem pro názorné vyzkoušení fyzikálního simulátoru.

Předposledním krokem je vložení několika modelů aut do scény. Je důležité dát pozor na to, aby vytvořené modely nebyly vytvořeny „přes sebe“ – tzn. Aby se neprolínaly. Po spuštění simulace by to mělo nečekané účinky. Nastavení modelu, který má sledovat kamera a který má být ovládán klávesnicí následuje.

Na závěr aplikace se nastaví počáteční pozice kamery a musí se spustit renderovací smyčka systému Coin3D.

## 5.3 Ovládání ukázkové aplikace

Po spuštění aplikace se zobrazí okno aplikace a v něm scéna, kterou jsme v programu nastavili. Kamera sleduje scénu z místa na modelu, který jsme definovali při tvorbě modelu. Pokud jsme vložili do scény modelů více, lze se mezi nimi přepínat klávesou **TAB**.

Modely aut se ovládají velice intuitivně, a to **šipkami** na klávesnici. Ukončení programu můžeme vynutit klávesou **ESC**, nebo kliknutím na křížek v pravém horním rohu okna.

## 5.4 Aplikace na přiloženém CD

Na přiloženém CD nalezneme několik verzí programu, tak jak postupoval vývoj modelu popsaný výše. Také přikládám aplikace vyvinuté v rámci předchozích projektů. Všechny aplikace jsou přiloženy ve spustitelném tvaru i ve formě zdrojových textů pro jejich případné další využití v budoucnu.

Pro sestavení je zapotřebí Microsoft Visual Studio .NET 2003. a nainstalované knihovny OPAL a Coin3D (použité verze k dispozici ke stažení z internetu na adresách [a] a [b] nebo také k dispozici na CD)

# 6 Závěr

Tato práce prokázala, že vytvoření simulace pohybu auta po nerovném povrchu ve 3D scéně za použití knihoven Coin3D pro zobrazení a knihovny OPAL pro fyzikální simulace je možné. Naopak si myslím, že vzájemná kompatibilita obou systémů je k tomu předurčuje. Důležité také je, že oba systémy jsou ve fázi neustálého vývoje a vylepšování, což z nich dělá perspektivní kandidáty pro toto použití.

Práce se simulátorem OPAL je možná místy nepřehledná a některé reakce simulátoru jsou nepředvídatelné, ale vytvoření této stabilní simulace auta může posloužit jako dobrý základ pro budování složitějších modelů a simulací na tomto principu do budoucna. Systémy OPAL i Coin3D přes všechny problémy nabízí solidní výkon a použitelnost ve všech jednodušších i profesionálních projektech.

Systém Coin3D je použit v několika projektech v tomto i nižších ročnících a všechny budou mít zřejmě stejnou či podobnou strukturu hlavní smyčky. V těch programech, kde je zapotřebí fyzikální simulace, muže použití knihovny OPAL místo používání vlastních, složitých a často méně přesných matematických výpočtů snížit požadavky na výpočetní výkon nebo zvýšit přesnost.

Na druhou stranu je nutné dávat při použití knihovny OPAL pozor na stabilitu vytvořeného modelu. V nejbližší době se očekává zveřejnění nové verze knihovny OPAL, která jistě některé nedostatky odstraní. Systém OPAL je opensource projekt a pracuje na jeho vývoji více lidí – dají se tedy očekávat nové verze i do budoucna.

# Literatura


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

[2] Coin3D, URL projektu: http://www.coin3d.org (květen 2006)

[3] Doc.Dr.Ing. Černocký, Jan, 5. přednáška z předmětu Základy počítačové grafiky: Transformace ve 2D/3D, k dispozici pro studenty FIT-VUT. URL: https://www.fit.vutbr.cz/study/courses/IZG/private (duben 2005)

[4] OPAL, URL projektu: http://opal.sourceforge.net (květen 2006)

[5] ODE, URL projektu: http://www.ode.org (květen 2006)

[6] Josie Wernecke, The inventor mentor, Addison-Wesley Proffesional, 1994, ISBN: 0201624958


# Další odkazy


[a] Coin3D verze 2.4.4 použitá v projektu k dispozici ke stažení zde:
ftp://ftp.coin3d.org/pub/coin/bin/win32//Coin-2.4.4-SoWin-1.3.0-bin-msvc7.zip (květen 2006)

[b] OPAL verze 0.3.1 použitá v projektu k dispozici ke stažení zde:
http://sourceforge.net/project/showfiles.php?group_id=119663&package_id=134434&release_id=364673 (květen 2006)