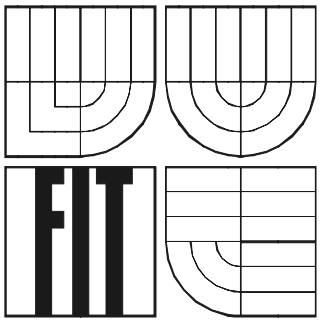
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Simulace vysoce náročného fyzikálního děje–urychlení elektronů v cyklotronu

Ročníkový projekt

2005 Milan Pindryč

Simulace vysoce náročného fyzikálního děje-urychlení elektronů v cyklotronu

Odevzdáno na Fakultě informačních technologií Vysokého učení technického v Brně, dne 11. května 2005.

© Milan Pindryč, 2005.

Autor díla převádí svá práva na reprodukci, distribuci a kopii celého díla i jeho části na Vysoké učení technické v Brně, Fakultu informačních technologií.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Pečivy
Další informace mi poskytl Rndr. Vojtěch Ullmann.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Milan Pindryč
3. května 2005

Poděkování :

Moje poděkování za odborné a organizační vedení a podporu při řešení problému patří především vedoucímu mého projektu **Ing. Janu Pečivovi.**

Abstrakt :

Tento projekt implementovaný v jazyce C++ s využitím nadstavbové knihovny nad openGL Open Inventor se zabývá trojrozměrnou simulací urychlování částic v kruhovém urychlovači, přesněji cyklotronu. Hlavním cílem práce bylo vytvořit co možná fyzikálně nejvěrohodnější model s ohledem na náročnost výpočtů tak, aby byl schopen fungovat v reálném čase. Velký důraz je kladen také na názornost urychlování a především na zobrazení trajektorie částic. Je možné měnit všechny parametry urychlovače za průběhu urychlení, což názorně demonstruje účinky a důsledky jednotlivých sil působících na částici v průběhu urychlení. Tato aplikace je určena primárně pro demonstraci účinků elektromagentických sil, nikoliv pro výpočet přesných hodnot rychlosti, proto je vhodná především jako pomůcka do hodin středoškolské fyziky.

Klíčová slova :

OpenGl, Open Inventor, cyklotron, kruhový urychlovač, trajektorie částic, reálný čas, elektromagnetické síly

Abstract

This project implemented in the C++ language with the use of library over OpenGL Open Inventor deals with 3D simulations of particle accelerators in the circular accelerator, precisely in the cyclotron. The main goal was to create possibly the most physically authentic model considering the difficulty of the calculations, which could work in real time. Great emphasis is also laid on the demonstration of the acceleration and mostly the display of particle trajectories. It is possible to change all parameters of the accelerator during acceleration which illustrates the effects and results of each of the forces affecting the particles in the course of acceleration. This application is primarily meant for demonstrating the affect of electromagnetic forces but not for the calculation of the exact figures of speed and as a result it is mostly used as studying material in high school physics.

.

Keywords

OpenGL, Open Inventor, cyclotron, particle trajectories, real time, electromagnetic forces.

Obsah

Seznam obrázků

1 Úvod

1.1 Fyzikální a matematická podstata urychlování částic

V této kapitole bych čtenáře rád seznámil s důvody pro urychlování částic a s jejich základními principy. Pravděpodobně se neobejdeme bez uvedení nezbytných vzorců a jejich odvození. čtenář se dozví, že existuje celá řada urychlovačů částic. Popíšeme si zde jejich výhody a nevýhody. Podrobněji se seznámí s principem kruhových urychlovačů a to hlavě cyklotronu, jehož simulace je předmětem této ročníkové práce. Tato kapitola vyžaduje od čtenáře dobré fyzikální znalosti z této oblasti fyziky, ale její pochopení je nevyhnutelné pro správné používání programu pro simulaci.

1.2 Prostředky využité pro implementaci

Zde bych rád podrobněji popsal použité prostředky pro tvorbu tohoto programu. Program je implementován v jazyce C++ s využitím nadstavbové knihovny nad openGL - implementace Coin jež je dílem norské společnosti System In Motion. Ta je volně dostupná pod licencí GPL a je plně kompatibilní s Open Inventorem. Podporovaných platforem pro tuto knihovnu je velmi mnoho. Z těch nejznámějších: Windows, Linux, Mac od Apple a obrovské množství Unixů. My se soustředíme pouze na Linux a Windows. Na Linuxu je použitelný překladač gcc, na Windows je funkční zatím pouze Microsoft Visual C++, nejlépe verzi 6, který je využit pro přeložení simulátoru a sice ve verzi Microsoft Visual C++ 6.0, Introductory Edition.

Nyní něco blíže k designu Open Inventoru. Je to knihovna napsaná v C++ a postavená nad OpenGL, která posunuje programátora od primitivního OpenGL rozhraní na vyšší úroveň a nabízí mu rozsáhlou množinu C++ tříd. Ta podstatně zjednodušuje práci programátora a dokonce často poskytuje vyšší výkon než přímá implementace v OpenGL. Vyšší výkon je možný díky jistým optimalizacím, které Open Inventor může provádět nad daty scény. Běžný programátor také obyčejně

nemá čas provádět profilování a optimalizaci renderovacích algoritmů. Proto již vyprofilované rutiny Inventoru nejsou špatnou volbou.

1.3 Tvorba 3D scény cyklotronu

V této části bude shrnuta a podrobně popsána celá podstata programu a to jak tvorba scény z jednotlivých komponent, tak i implementace urychlení částice. Převážnou část věnuji mým algoritmům použitým pro tvorbu scény nebudu se zde příliš zabývat popisem vlastní knihovny Open Inventor. Největší důraz bude kladen na implementaci pohybu částice v proměnlivém elektromagnetickém poli což tvoří jádro simulace.

1.4 Ovládání urychlovače

Zde bych rád popsal tvorbu jednoduchého uživatelského menu pomocí knihovny Open inventor. Rovněž se zde zaměřím na popis ovládání celého programu. A vysvětlím funkce asociované k jednotlivým položkám v menu. Popíši jednotlivé zobrazované hodnoty jako například intenzitu magnetického pole, intenzitu elektrického pole, sfázovanost frekvencí oběhu a frekvence elektrického pole a další.

1.5 Co bude v závěru

Závěrečná kapitola bude obsahovat zhodnocení dosažených výsledků . Objeví zde i zhodnocení z pohledu dalšího vývoje projektu a zhodnocení jeho nedostatků pokusím se zde srovnat svůj projekt s aplikací řešící podobný problém ale jen ve dvourozměrném prostředí.

2 Fyzika urychlování částic

2.1 Proč používat urychlovače částic

Pro studium vlastností, struktury a interakcí elementárních částic, jakož i pro aplikace v různých oblastech vědy a techniky (včetně medicíny), je potřeba použít částic urychlených na vysoké kinetické energie. Uměle urychlit dovedeme pouze stabilní elektricky nabité částice - elektrony e^-, pozitrony e^+ , protony p^+, deuterony d^+, ionty hélia He^{++}=a-částice a ionty těžších prvků. Vysokoenergetické částice bez náboje (jako jsou fotony g, neutrony n^o, neutrální piony, ...) a krátkožijící částice (p-mezony, hyperony, ...) lze pak získat sekundárně - interakcemi urychlených nabitých částic s částicemi ve vhodném terčíku. Přístroje, které působením silných elektrických a magnetických polí urychlují nabité částice, se nazývají urychlovače. Podle způsobu technické realizace a tvaru dráhy, na níž urychlování částic probíhá, rozdělujeme urychlovače na dva základní typy: lineární **a** kruhové**.**

2.2 Součásti urychlovačů částic

Než se budeme zabývat jednotlivými typy urychlovačů, zmíníme se o dvou součástech, které mají všechny urychlovače - zdroj urychlovaných částic a terčík.

2.2.1 Terčík

Terčík, na nějž dopadá svazek urychlených částic, je buď vnitřní - je umístěn uvnitř urychlovacího systému, nebo vnější - svazek částic je vyveden ven z urychlovací trubice. Rovněž sekundární částice, produkované na vnitřním terčíku (jako jsou p nebo K mesony), se působením magnetického a elektrického pole vyvádějí ve formě svazku do prostoru laboratoře, kde jsou umístěny měřící aparatury (detekční přístroje, bublinové komory atd.). Při dopadu urychlených částic na terčík se většina kinetické energie částic mění na teplo - ostřelovaný terčík se zahřívá. Aby nedošlo k jeho tepelnému poškození či odpaření terčíkové látky, je nutno toto ztrátové teplo (může činit i stovky wattů) odvádět - terčík se fixuje na masívní kovovou podložku s dutinou, chlazenou protékající vodou (podobně jako anody výkonových rentgenových trubic).

2.2.2 Iontový zdroj částic

Zdroj urychlovaných částic emituje do "startovacího" místa urychlovacího systému požadovaný druh částic, jako jsou elektrony, protony či těžší ionty. V nejjednodušším případě se jedná o ionizační trubici obsahující příslušný zředěný plyn (např. vodík H), kde v doutnavém výboji mezi katodou a anodou (při napětí cca stovky voltů) vznikají ionty (u vodíku jsou to protony p^+) a ty jsou pomocí tenké kapiláry vedeny "odsávací" elektrodou do urychlovacího systému. Pro urychlovače elektronů je zdrojem prostá žhavená katoda (termoemise elektronů) opatřená vhodnými urychlujícími a fokusujícími anodami - "elektronovým dělem" - podobně jako u obrazovky. U velkých urychlovačů vysokých energií se jako zdroje částic k urychlení někdy používají injektory - do hlavní komory jsou částice vstřikovány pomocným lineárním urychlovačem (s energií jednotky až desítky MeV) a následně urychlovány na požadovanou vysokou energii (GeV).

2.3 Rozdělení urychlovačů a jejich perspektivy

2.3.1 Lineární urychlovače

Lineární urychlovače urychlují nabité částice působením elektrického pole během jejich pohybu po lineární přímkové dráze. Můžeme je rozdělit na elektrostatické (vysokonapěťové) a vysokofrekvenční. Jejich princepem je přidávání energie elektronu na přímkové dráze pomocí vysokého napětí. Jejich hlavní nevýhodou jsou obří rozměry.

2.3.2 Kruhové urychlovače

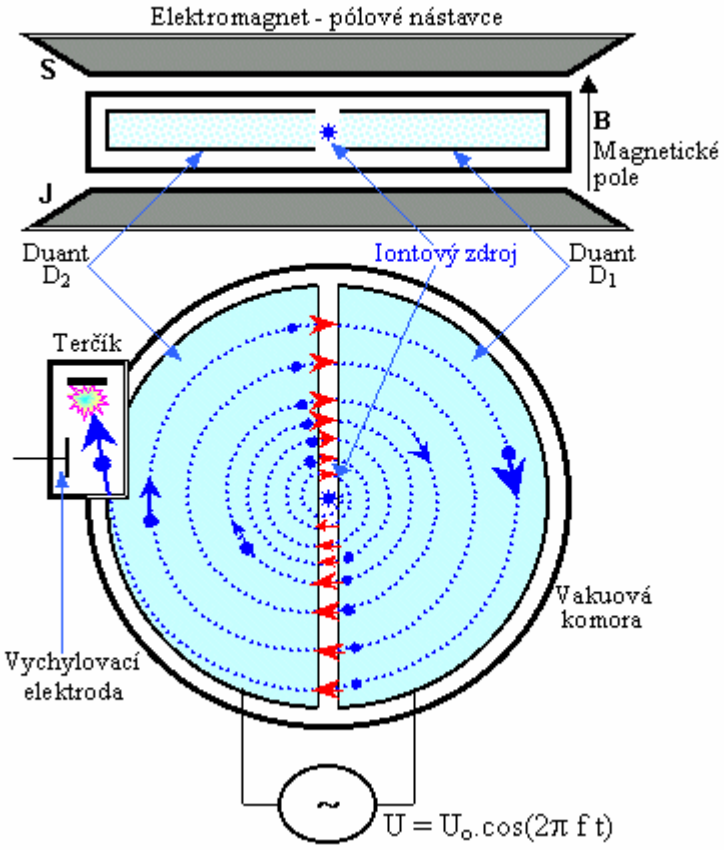
Velmi efektivním způsobem, jak urychlit nabité částice na vysoké energie, je jejich mnohonásobné urychlení v elektrickém poli, kam jsou částice opakovaně vraceny po kruhové dráze působením magnetického pole. Na částici s nábojem q je zde aplikována nejen elektrická urychlující síla Fe = q.E, ale i Lorentzova síla Fm = q.[v ´ B] působící v magnetickém poli intenzity B kolmo ke směru pohybu nabité částice rychlostí v. Tato magnetická síla způsobuje, že nabitá částice se bude pohybovat po kruhové dráze o poloměru R = m.v.c/(q.B). Je-li ve vhodných místech této kruhové

dráhy synchronně aplikováno elektrické urychlující pole (v tečném směru), budou částice periodicky urychlovány při každém svém oběhu.

2.3.2.1 Cyklotron

Základním typem kruhového urychlovače je cyklotron (první cyklotron vyvinul E.O.Lawrenc již v r.1932), jehož princip je schématicky znázorněn v obr.1

Obr.1:



Mezi póly silného elektromagnetu jsou v ploché vakuové komoře upevněny dva duté poloválce D1 a D2, tzv. duanty, mezi nimiž je urychlovací mezera. Duanty jsou připojeny ke zdroji střídavého napětí U = Uo.cos(2pf.t) o frekvenci f (bývá kolem 20MHz), takže v mezeře mezi deskami je střídavé elektrické pole. Nabité částice vstupují do středu urychlovací mezery z iontového zdroje]. Následkem síly, kterou elektrické pole v mezeře působí na částici s nábojem q a hmotností m, je částice vtažena do jednoho z duantů (který má právě opačnou polaritu) s určitou rychlostí v1. Uvnitř duantu, kde je elektrické pole odstíněno, působením silného magnetického pole B opíše částice půlkružnici o poloměru R1 = m.v1/(q.B) (tento poloměr je dán rovnováhou mezi odstředivou silou a Lorentzovou magnetickou silou: $m.v1^2/R1 = q.B.v1$). Doba, za kterou projde částice tuto půlkružnici,

je T = pR1/v1 = pm/(q.B) - vidíme, že tato doba (půl-perioda) oběhu částice nezávisí na její rychlosti ani na jejím poloměru dráhy; frekvence kruhového oběhu částice tedy je f = q.B/(2pm) a je konstantní, protože m, q a B jsou v daném uspořádání konstanty. Jestliže jsou duanty napájeny střídavým napětím právě o této frekvenci f (je splněna podmínka rezonance či synchronizace), pak v okamžiku kdy částice opíše půlkružnici v prvním duantu a ocitne se opět v urychlovací mezeře, je polarita duantů již opačná a částice bude opět urychlena elektrickým polem, takže do druhého duantu vletí s větší rychlostí v2>v1. V druhém duantu se bude pohybovat opět po kružnici, nyní však o poloměru R2 = m.v2/(q.B), který je větší než byl R1, ale se stejnou periodou a frekvencí kruhového pohybu.

Stejným způsobem je pak částice při každém svém průchodu mezerou mezi duanty znovu a znovu urychlována, přičemž se pohybuje po kružnicích s rostoucím poloměrem, tedy po spirále (obr.1.5....). Z poslední své dráhy o maximálním poloměru (blízkém poloměru duantů) je urychlená částice elektrostaticky nebo magneticky vychýlena a vyvedena do prostoru terčíku, na nějž narazí a vyvolá tam jaderné procesy.

Nastíněný princip činnosti cyklotronu bude při konstantní frekvenci fungovat jen do té doby, kdy hmotnost urychlované částice můžeme považovat za konstantní, tj. pouze v nerelativistické oblasti. Chceme-li použít cyklotronu k urychlování částic na vyšší energie, kdy rychlost částic je již srovnatelná s rychlostí světla, přestává být hmotnost částice m konstantní, ale zvyšuje se s rostoucí rychlostí: $m = mo/Ö(1-v^2/c^2)$. Ve stejném tempu se snižuje frekvence oběhu částic v konstantním magnetickém poli: $R = mo.v/(q.B.Ö(1-v^2/c^2))$. Aby mohla být částice dále urychlována i v této relativistické oblasti, je potřeba modulovat frekvenci urychlovacího napětí tak, aby byla stále v rezonanci s frekvencí oběhu částice. Takto upravený cyklotron se "synchronizací" se nazývá synchrocyklotron nebo relativistický cyklotron (ve starší literatuře se vyskytuje i název "fázotron"). Tyto přístroje pracují v pulsním režimu, přičemž kmitočet urychlovacího napětí na duantech je modulován a mění se cca 50-krát za vteřinu z hodnot cca 25MHz na cca 12MHz, používají se pro urychlování protonů na energie do cca 1GeV.

2.3.2.2 Synchrotron

Pro urychlování částic na velmi vysoké energie vychází v kruhovém urychlovači velký poloměr jejich orbit, takže cyklotronový způsob se spirálovým pohybem částic v ploché vakuové komoře již není prakticky použitelný. Aby dokonale vakuový prostor nebyl enormně velký, stejně jako elektromagnety, je nutno použít kruhové urychlovače s pevnou kruhovou drahou. Aby se nabitá částice urychlovala a udržela se na pevné kruhové dráze o poloměru R, je potřeba aby s rostoucí rychlostí v(t) urychlovaných částic se s časem synchronně zvyšovala jak frekvence f(t) urychlovacího

napětí, tak intenzita magnetického pole B(t), která již nemůže být konstantní, ale je rovněž funkcí času.

2.3.2.3 Betatron

Urychlovací trubice betatronu má tvar prstence (toroidu) zhotoveného z elektricky nevodivého materiálu (sklo, porcelán) s vysokým vakuem uvnitř. Trubice je umístěna ("navléknuta") mezi pólovými nástavci elektromagnetu, napájeného střídavým proudem. Elektrony jsou ve vhodném okamžiku (vhodné fázi periody střídavého proudu) vstřikovány do urychlovací trubice elektronovou tryskou, tvořenou žhavenou katodou, mřížkou a urychlující a fokusující anodou - je to pobobné "elektronové dělo" jako je u obrazovky. Časově proměnné magnetické pole indukuje v trubici vířivé elektrické pole, jehož elektromotorická síla, směřující podél kruhové dráhy, tyto elektrony urychluje.

2.3.3 Perspektivy urychlovačů

Jakkoli je princip kruhového urychlování nabitých částic velmi úspěšný a efektivní, zdá se, že kruhové urychlovače se již přiblížily k hranicím svých možností. Pokud bychom chtěli nabité částice urychlovat na ještě podstatně vyšší energie při reálně dostupných průměrech kruhové dráhy (tj. průměrech urychlovacích trubic), čím dál více by se uplatňoval jev vzniku synchrotronového záření, které by odnášelo značnou část kinetické energie částic a nakonec by znemožnilo další urychlení. Zdá se tedy, že budoucí urychlovače pro nejvyšší energie budou muset být lineární.

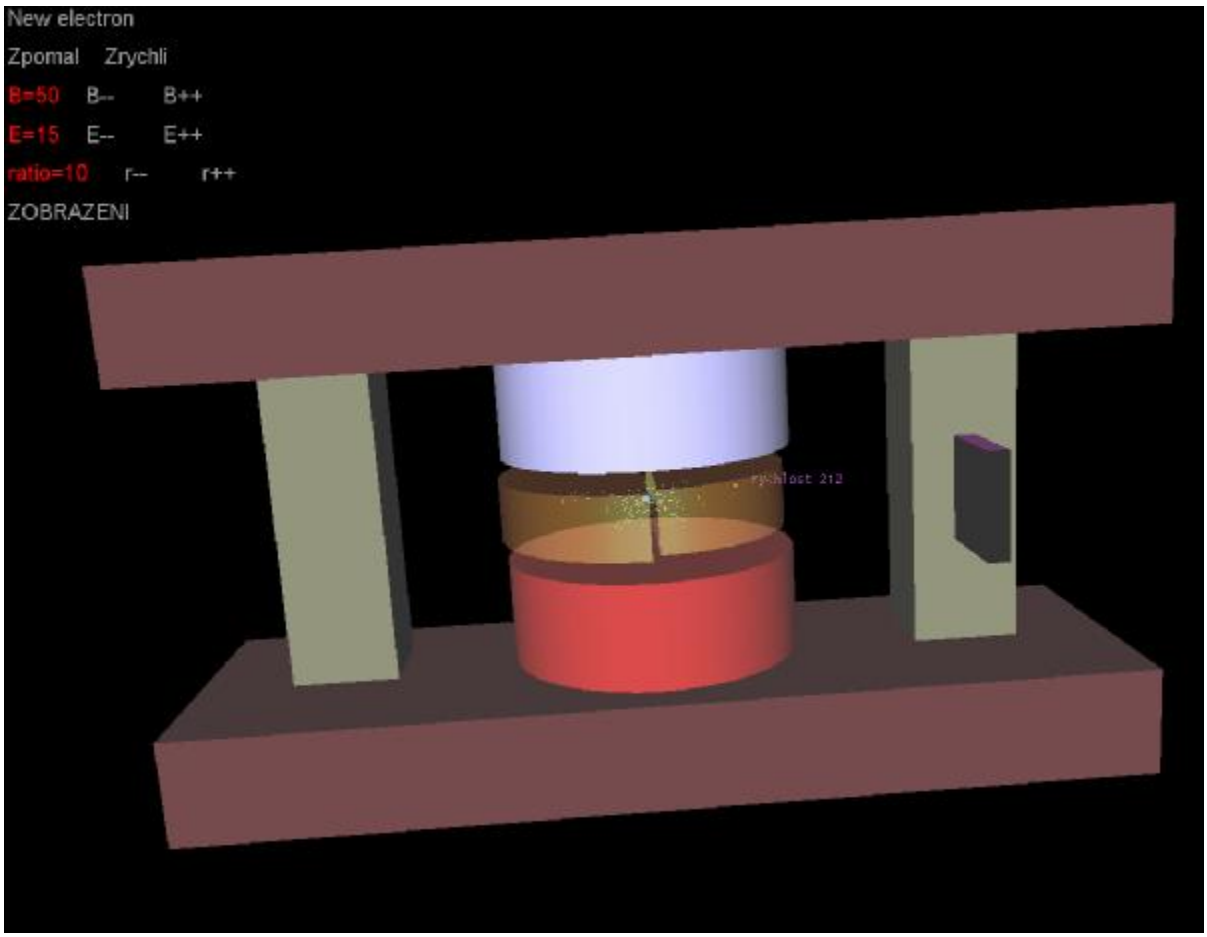
3 Scéna urychlovače

3.1 Úvodem

Design Open Inventoru vychází z konceptu grafu scény. Tedy, scéna je složena z uzlů - anglicky nodes. Nody jsou různých typů. Jedny nesou informace o geometrii těles (krychle, kužel, model tělesa), další různé atributy (barva, textury, souřadnice objektu) a také existují speciální nody, které obsahují seznam jiných nodů, anglicky zvané groups. A právě tyto groupy umožňují organizovat ostatní nody do hierarchických struktur zvaných grafy. Takovýto graf nám pak reprezentuje naši scénu.

3.2 Skladba scény urychlovače

Obr.2:



3.2.1 Jednoduché objekty

Scéna se skládá především z jednoduchých objěktů které poskytuje knihovna Open Inventor. Tyto oběkty tvoří nepodstatnoů část simulace. Jsou to dva magnery které jsou vytvořeny pomocí funkce využívající třídy SoCilinder. Modrý je severní je natočen severním pólem do středu a červený jižním

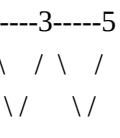
Další součástí je emitor částic – elektronů ,který je umístěn uprostřed mezi dvěma duanty a je opět tvořen stejnou funkcí jako magnety.

Na obrázku 2 je dále dobře patrná nosná konstrukce celého urychlovače a terčík pro dopad elektronů, který je znázorněn temě fialovou barvou. Scéna záměrně nevyužívá textury ani další světla protože podstatou je rychlost a názornost simulace nikoliv dokonalý vzhled. Textury byly původně použity ale celou scénu značně znepřehlednovali. Vlevém horním rohu je vidět menu kterému bude věnována samostatná kapitola.

3.2.2 Oběkty vytvářené z trojúhelníků

Oběkty vytvářené z trojůhelníků jsou v mém případě duanty-duté půlválce, které jsou tvořeny ze dvou pulkružmic a z jednoho pulkulateho pásku(půl obruče). Nejjednodušší jak genrovat trojuhelníky je GL_TRIANGLES, kdy specifikujeme vždy tři body, které budeme nazývat vertexy. Každý z těchto vertexů se skládá ze třech prostorových souřadnic - x,y,z. Tyto tři vertexy tedy pošleme do OpenGL a ono nám vyrendruje jeden trojúhelník. Většina trojúhelníků sdílí vrcholy se svými sousedy, proto byly vymyšleny věci jako GL_TRIANGLE_STRIP, který vždy z prvních třech vertexů vykreslí první trojúhelník, ale od té chvíle již vykresluje trojúhelník s každým dalším vertexem, přičemž chybějící dva vertexy použije z předchozího trojúhelníku. Vše demonstruje obr.3.

Obr.3:



Kromě velmi ojedinělých případů bude použití GL_TRIANGLE_STRIP vždy rychlejší než obyčejné trojúhelníky, neboť se do OpenGL posílá méně dat pro vykonání stejné práce. Z toho vyšli i designéři Open Inventoru a navrhli dvě třídy pro rendrování trojúhelníků: SoTriangleStripSet a SoIndexedTriangleStripSet. Niže uvádím zdrojové kódy funkcí využívajících třídu SotriangelStripSet. Nejdříve se nageneruje seznam řídících vrcholů a potom se znich vytvoří trojúhelníky.

Zdrojový kód pro půlobruč:

```
float c[detail_valec];
float d[detail_valec];
int i;
for(i=0;i<(detail_valec-1);i++)
  {
  float b;
  b=180/(detail_valec-1)*M_PI/180*i;
  c[i]=sin(b);
  d[i]=cos(b);
  }
c[detail_valec-1]=0;
d[detail_valec-1]=-1;

float vertices[detail_valec*2][3];
int j;
for(j=0;j<(detail_valec*2);j++)
{
vertices[j][0]=c[j / 2]*radius;
vertices[j][1]=(j % 2)*hight;
```

```
vertices[j][2]=d[j / 2]*radius;
}
 SoCoordinate3 *coords = new SoCoordinate3;
 coords->point.setValues(0,detail_valec*2, vertices);
 root->addChild(coords);

SoTriangleStripSet *strip = new SoTriangleStripSet;
strip->numVertices.set1Value(0, detail_valec*2);
root->addChild(strip);
```

Zdrojový kód pro půlkružnici:

```
float c[detail_valec];
float d[detail_valec];
int i;
for(i=0;i<detail_valec-1;i++)
  {
  float b;
  b=180/(detail_valec-1)*M_PI/180*i;
  c[i]=sin(b);
  d[i]=cos(b);
  }
c[detail_valec-1]=0;
d[detail_valec-1]=-1;


float vertices[detail_valec][3];
int k;
for(k=0;k<detail_valec;k++)
  {
  if ((k%2)==0){
            vertices[k][0]=c[(k+1)/2]*radius;
            vertices[k][1]=0;
            vertices[k][2]=d[(k+1)/2]*radius;
            }
  else
            {
            vertices[k][0]=c[detail_valec-(k+1)/2]*radius;
            vertices[k][1]=0;
            vertices[k][2]=d[detail_valec-(k+1)/2]*radius;
            }
  }
 SoCoordinate3 *coords = new SoCoordinate3;
  coords->point.setValues(0,detail_valec, vertices);
  root->addChild(coords);

SoTriangleStripSet *strip = new SoTriangleStripSet;

 strip->numVertices.set1Value(0, detail_valec);
root->addChild(strip);
```

3.2.3 Elektron

Pro elektron je vytvořena vlastní třída ELEKTRON která uchovává potřebné informace o elektronu, jako jsou souřadnice na kterých se nachází, x-y-z-složka rychlosti, a několik příznaků potřebných pro urychlování. Všechny elektrony se ukládají do vectoru elektronů. Konstruktor nastavuje všechny potřebné hodnoty do výchozích pozic a generuje náhodnou počáteční rychlost elektronu.

Elektron má dále důležitou funkci setPosition, která tvoří jádro celého urychlovače. Počítá totiž pozici elektronu podle okolí elektronu více se o ni zmíníme v následující kapitole. Další velmi důležitou funkcí je statická členská funkce updatePosition která projde všechny elektrony a zavolá pro ně funkci setPosition.

Graficky je elektron reprezentován koulí s náhodnou barvou která se mu přiděluje při jeho vytvoření. K vytvoření koule je použita třída SoSphere.

3.3 Funkčnost scény urychlovače

Máme vytvořenu třídu elektronů kde budou uchovány všechny podstatné informace (poloha, hmotnost, vektor rychlosti a ukazatel na node *SoTranslation* zajišťující přesun planety na správné místo). Dále máme funkci která zajistí pepočítání pozic elektronů a potřebujeme ji jednou začas zavolat. Právě k tomu se hodí senzory.Senzory jsou objekty, které sledují určité události a v případě jejich výskytu zavolají programátorem vytvořenou funkci. Mezi možné sledované události patří změna hodnoty pole (*SoFieldSensor*), uplynutí časového kvanta (*SoAlarmSensor*, *SoTimerSensor*), nečinnost aplikace (*SoIdleSensor*) a další.Do scény je přidán *SoOneShotSensor*. Prvním parametrem v konstruktoru je ukazatel na callback funkci, druhým jsou data jí předávaná (typ *void **).

3.3.1 Callback funkce

Ve funkci sensorCallback zjistíme čas uplynulý od jejího minulého volání, provedeme výpočet nových pozic elektronů (tučně ve zdrojovém kódu) a metodou schedule() opět zařadíme senzor do fronty (v proměnné sensor je ukazatel na senzor, který funkci zavolal). Tím máme zajištěno, že bude dosaženo maximálního možného počtu vyrenderovaných snímků za sekundu.

Zdrojový kód callback funkce:

SbTime dt;

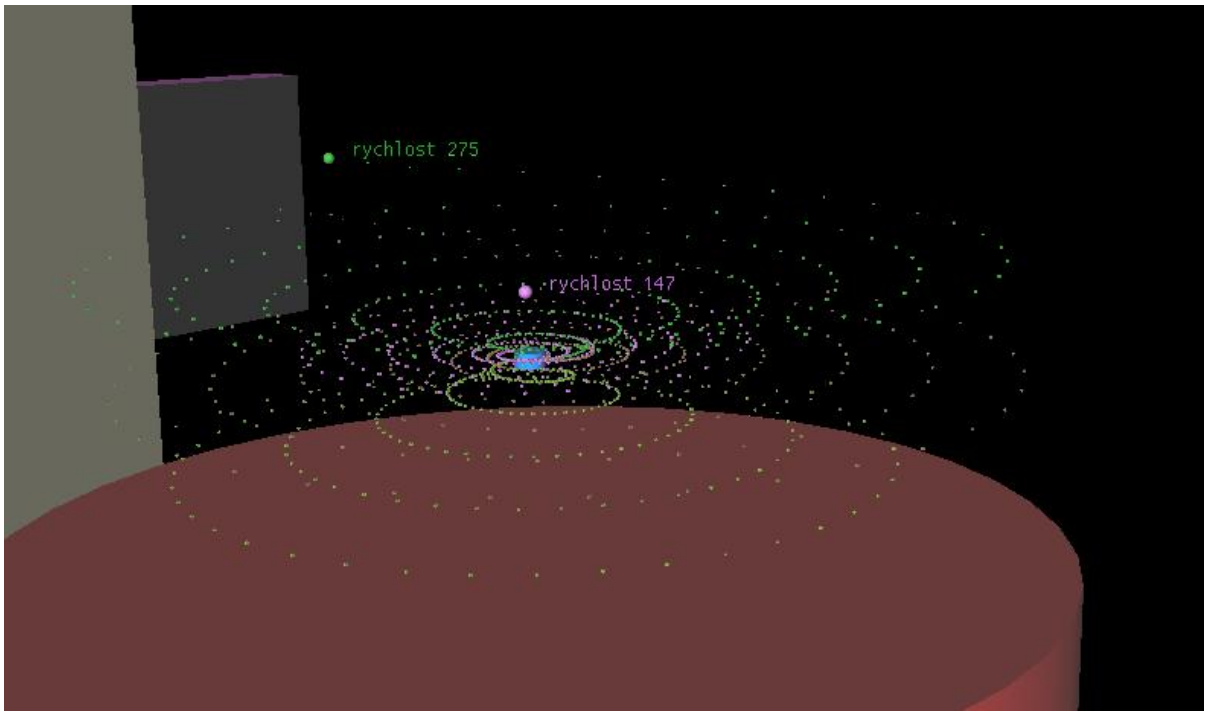
```
SbTime cas_zacatku=SbTime::getTimeOfDay();

void sensorCallback(void *, SoSensor *sensor) {
SbTime dt, ct = SbTime::getTimeOfDay();
if (lastTimeTick == SbTime::zero()) {
        lastTimeTick = ct;
        sensor->schedule();
        return;
  }
  dt = ct - lastTimeTick;
  lastTimeTick = ct;
  float E = E_max*cos(frekvence*(cas_zacatku.getValue() - ct.getValue())/zpomaleni);
  elektron::updatePositions(E,elektrons,dt/zpomaleni,numelektrons);
  sensor->schedule();
}
```

3.3.2 Rovnice pro výpočet pozic a rychlostí elektronů

Tvoří nejduleřitější část programu. Počítá vektorově podle rovnic uvedených v druhé kapitole síly působící na elektron. Z těchto sil potom vypočítává zrychlení jednak dostředivé, udržující elektrony v duantech na kruhových drahách, které způsobuje magnetická síla a také zrychlení způsobované elektrickým polem mezi duanty tvořící přírustek absolutní hodnoty rychlosti. Duůežitým parametrem této vunkce je delta času tj. čas co uplynul od posledního volání callback funkce. Jelikož rovnice použité pro výpočet jsou v diferenciální a numerický výpočet platí jen pro malé delta je nutno počítat pozici výrazne čatěji než je volána callback funkce a to řeší cyklus for vněmž se pozice počítá asi 500 krát přesněji.

Další důležitá věc, kterou tato funkce řeší, je kolize se stěnami urychlovače.

Obr.4:



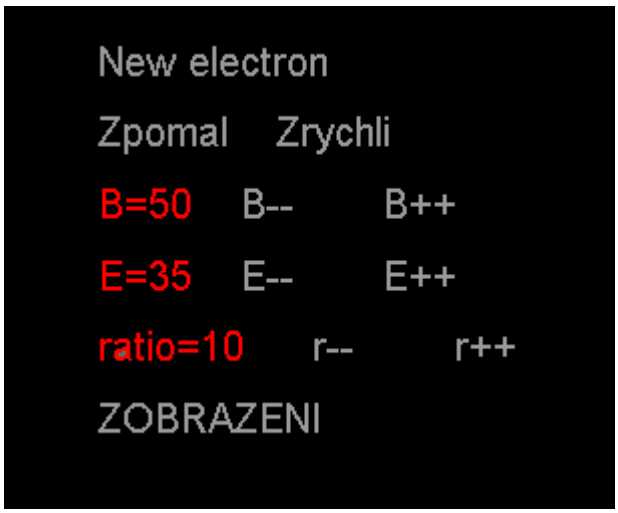
Zdrojový kód pro výpčet pozice:

```
void elektron::setPosition(float E,float delta_time,vector<elektron> &elektrons) {
float flag3=0;
float d_speed[3];
string st;
st=" rychlost "+stringify(sqrt(speed_x*speed_x+speed_z*speed_z)*10);
btn1->string=st.c_str();

        for (int lt=0;lt<presnost;lt++){
        float pom_d_speed=0;

                if((a_x*Flag)<=0){
                        Flag=-Flag;
                        flag3=1;
                        pom_d_speed=q_castice/m_castice*E;
                                }


  d_speed[0]=pom_d_speed+((speed_z)*B*q_castice/m_castice*delta_time/presnost);
                        d_speed[1]=0;
                        d_speed[2]=-((speed_x)*B*q_castice/m_castice*delta_time/presnost);
                        if ((a_z<-6) && flag3){Flag2=0;}//konec urychlování

                        if (Flag2==1)
                        {a_x = a_x+d_speed[0]*delta_time/presnost+speed_x*delta_time/presnost;
                        a_y = a_y+d_speed[1]*delta_time/presnost+speed_y*delta_time/presnost;
                        a_z = a_z+d_speed[2]*delta_time/presnost+speed_z*delta_time/presnost;
```

```
                    speed_x = speed_x+d_speed[0];
                    speed_y = speed_y+d_speed[1];
                    speed_z = speed_z+d_speed[2];
                }
                else{
                    a_x = a_x+speed_x*delta_time/presnost;
                    a_y = a_y+speed_y*delta_time/presnost;
                    a_z = a_z+speed_z*delta_time/presnost;
                    }
                if (a_x<-15){
                    smazani elektronu(); //trfil terčík
                    }

        }

if ((((sqrt(a_z*a_z+a_x*a_x)<=-7)||
(sqrt(a_z*a_z+a_x*a_x)>=7)) && (Flag2))||
(a_y<-1.5)||(a_y>1.5))                          //kolize se stenou urychlovace
  {
                    smazani elektronu();    //po kolizi se stěnou urychlovače
  }
```

3.4 Ovládání urychlovače

Obr.5:



Ovládání urychlovače je tvořeno pouze tlačítky což je pro náš účel dostačující. Tlačítko *New Elektron* vytvoří nový elektron. Tlačítka *Zpomal* a *Zrychli* umožňují nastavování rychlosti plynutí času. Zobrazení odstraňuje ze scény část urychlovače aby bylo možné lépe sledovat trajektorie. Důležité je hodnota ratio která ukazuje shodu fáze elektrického pole a doby oběhu k urychlení dochází nejlépe u ratio = 10. B je magnetická indukce a E je intenzita elektrického pole.

3.4.1 Vytvoření menu

Menu je tvořeno nápisy třídy SoText2(), které jsou promítnuty na obrazovku za použití ortografické kamery. Díky přepočtu souřadnic se menu drží na jednom místě a nedochází k jeho zkreslení.

Zdrojový kód pro vytvoření menu s jedním tlačítkem:

```
SoSeparator * MakeMenu(SoWinExaminerViewer * viewer){
SoSeparator * MyMenu = new SoSeparator;

SoCallback *overlayCB = new SoCallback;
overlayCB->setCallback(overlayViewportCB);
MyMenu->addChild(overlayCB);

camera = new SoOrthographicCamera;
MyMenu->addChild(camera);

SoEventCallback * ecb = new SoEventCallback; //callback udalosti pro mys
ecb->addEventCallback(SoMouseButtonEvent::getClassTypeId(), event_cb, viewer);
MyMenu->addChild(ecb);

SoMaterial *material0 = new SoMaterial;
material0->diffuseColor.setValue(SbColor(0.6f, 0.6f, 0.6f));
MyMenu->addChild(material0);

SoFont *myFont = new SoFont;
 myFont->name.setValue("Arial");
 myFont->size.setValue(20);
 MyMenu->addChild(myFont);

//novej elektron

SoText2 * b0 = new SoText2();
b0->string = "New electron";

MyMenu->addChild(b0);
```

3.4.2 Callback funkce

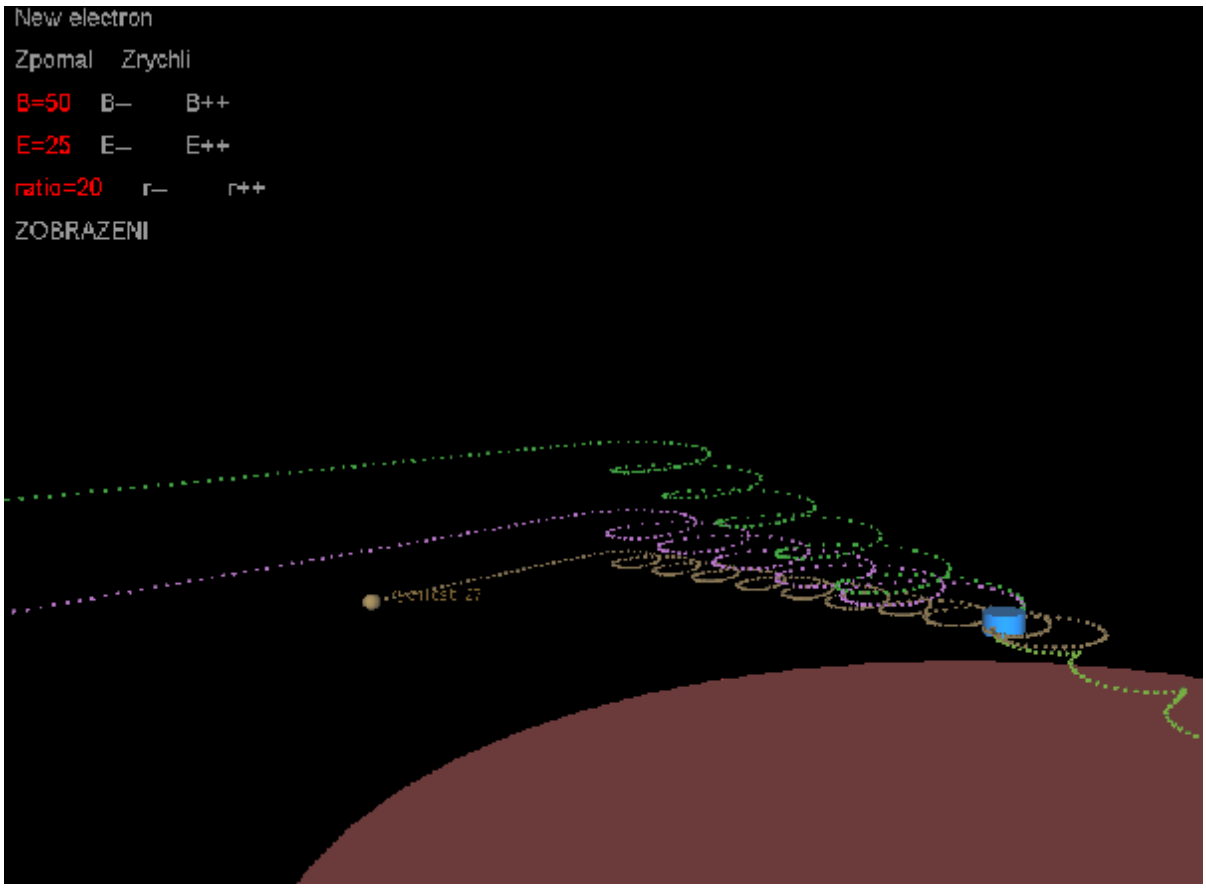
Na zachytávání událostí z periferních zařízení (myš,klávesnice) se používají třídy Open Inventoru. Třída SoEventCallback je k tomu, aby zachytila jakoukoliv událost. Metodě SoEventCallback::addEventCallback() předává jako první parametr typ události, na který má callback funkce event_cb() reagovat. Jméno callback funkce je uvedeno jako druhý parametr. V callback funkci nejdříve zjistíme jestli bylo stisknuto levé tlačítko (pravé nepoužijeme). Pro určení jaký objekt byl ve scéně myší vybrán se používá třída SoRayPickAction. Z informací o události zjistíme souřadnice kliknutí myši a ty předáme prostřednictvím metody SoRayPickAction::setPoint() instanci třídy SoRayPickAction. Projdeme s pomocí metody SoRayPickAction::apply() celý graf scény a zjistíme vybraný bod v prostoru. Jestliže byl takový bod nalezen, stačí jen ověřit, ke kterému objektu ve scéně patří.

Zdrojový kód callback funkce:

```
static void  event_cb(void * ud, SoEventCallback * n)
{
const SoMouseButtonEvent * mbe = (SoMouseButtonEvent *)n->getEvent();
if (mbe->getButton() == SoMouseButtonEvent::BUTTON1 &&
mbe->getState() == SoButtonEvent::DOWN) {
SoWinExaminerViewer * viewer = (SoWinExaminerViewer *)ud;

SoRayPickAction rp(viewer->getViewportRegion());
rp.setPoint(mbe->getPosition());
rp.apply(viewer->getSceneManager()->getSceneGraph());
SoPickedPoint * point = rp.getPickedPoint();
if (point == NULL) {   return;   }
 SoNode * MyNode = point->getPath()->getTail(); //jaky objekt se vybral
 SoText2  * pickedBtn; //sem si ulozim vybrane tlacitko/elektron
 n->setHandled();
 if (MyNode->getTypeId() == SoText2::getClassTypeId()){ //je to tlacitko
        pickedBtn = (SoText2 *) MyNode;
} else return;
```

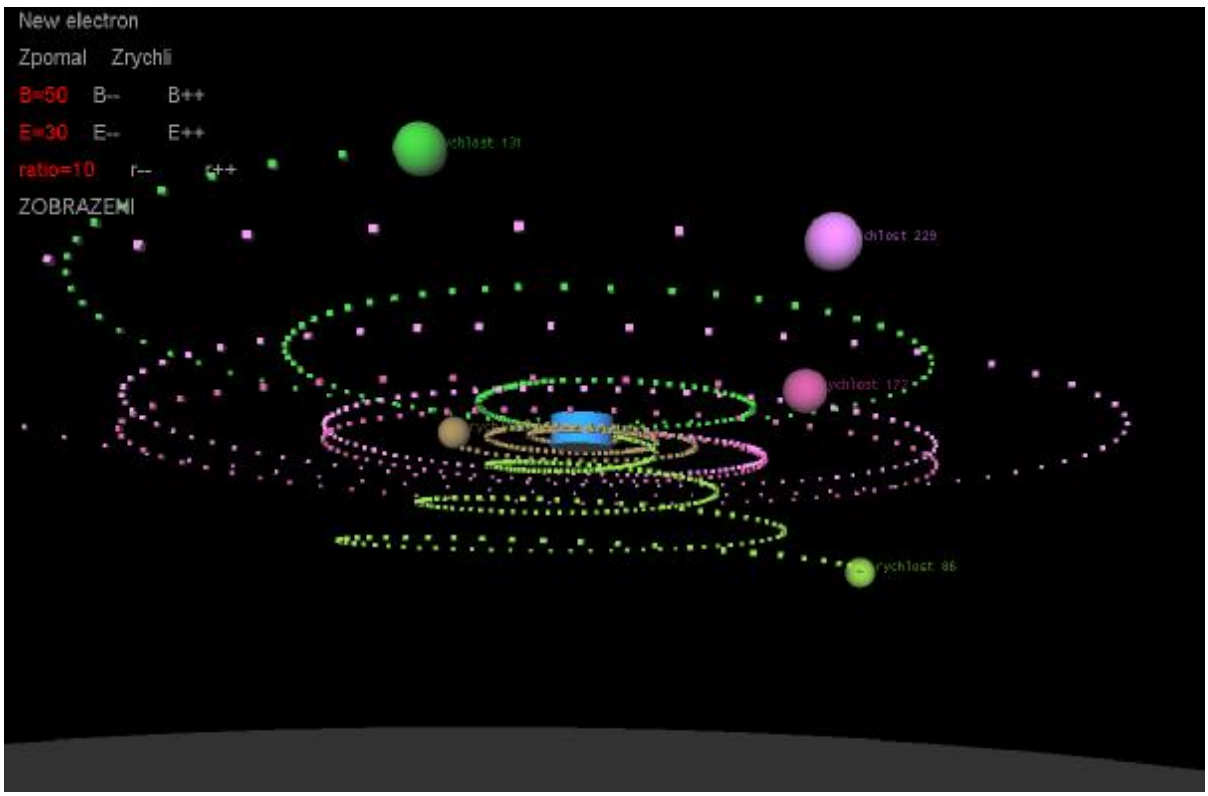
Nevhodně zvolené ratio - elektrony se neurychlují, ale dosáhli s nízkou rychlosí místa pro opuštění urychlovače:

Obr.6:



Vhodně zvolené ratio - elektrony se urychlují a což ukazují jejich spirálové dráhy.

Obr.7:



4 Závěr

Práce svým charakterem simulace a tudíž její přesností odpovídá původně zamýšlenému účelu použití a sice jako pomůcka vhodná pro použití například na středních školách.

Dovolím si srovnání se simulátorem http://www.phy.ntnu.edu.tw/java/cyclotron/cyclotron.html který je bohužel omezen na dvourozměrný prostor. Dále zde také, na rozdíl od mého simulátoru, není trajektorije počítána na základě silového působení, ale jedná se pouze o vykreslování půlkružnic, což neodpovídá realitě. Přírůstek rychlosti je zde rovněž řešen skokově. Moje pojetí lépe postihuje realitu.

Jsem si ale vědom několika nedostatků a znich vyplývajících možností vylepšení této práce. Tyto nedostatky jsou hlavně z oblasti výkonosti aplikace a to hlavně vykreslování trajektorie, které je velmi náročné protože s každým vykresleným bodem přibývá do scény 12 tojuhelníků což ji výrazně zpomaluje. Možnou optimalizací mohlo být dávání si krychliček pod separátor a až jich bude určitý počet (třeba 40), pak separátoru nastavit renderCaching na ON a dále už do něj nic nepřidávat. Vytvořit si další separátor a pokračovat dále stejným způsobem. Jinnou cestou by bylo konstruovat si krychličky v SoCoordinate a SoIndexedTriangleStripSet. Na dobrých grafických kartách by to mohlo přinést zlepšení. Tyto optimalizace by však byly na delší ladění a zkoumání co pomůže více.

Doufám že se mi ještě podaří na tuto práci navázat dovést ji k uceleněší aplkaci například s možností srovnání více druhů urychlovačů. Dále by bylo vhodné zpřesnění výpočtů rychlostí i s uvažováním vzájemných interakcí elektronů a v neposlední ředě zlepšení výkonnosti.

Literatura

[1] Halliday, D. – Resnick, R. – Walker, J. (2000): Fyzika, Brno, Vutium, Prometheus.

[2]The Inventor Mentor:Programming ObjectÿOriented 3D Graphics with Open Inventor, Release 2

http://root.cz/clanky/open-inventor/.

http://sweb.cz/AstroNuklFyzika/JadRadFyzika5.htm#Urychlovace