VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

VÝVOJ APLIKACÍ S VYUŽITÍM GRAFICKÉHO VÝVOJOVÉHO PROSTŘEDÍ IRRLICHT ENGINE

APPLICATION DEVELOPMENT USING IRRLICHT ENGINE

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — DANIEL BALCÁREK
AUTHOR

VEDOUCÍ PRÁCE — ING. JAN ROUPEC, PH.D.
SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2012/13

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Daniel Balcárek

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Vývoj aplikací s využitrím grafického vývojového prostředí Irrlicht Engine

v anglickém jazyce:

Application Development Using Irrlicht Engine

Stručná charakteristika problematiky úkolu:

Autor se seznámí s 3D grafickým vývojovým prostředím Irrlicht Engine a vyvine ukázkové aplikace.

Cíle bakalářské práce:

Cílem práce je vývoj ukázkových aplikací využívajících prostředí Irrlicht Engine. Vyvinuté aplikace budou tématicky zaměřené k propagaci studia na ÚAI.

Seznam odborné literatury:

Kyaw Situ A., Stein J.: Irrlicht 1.7 Realtime 3D Engine. Pack Publishing, London, 2011.

Vedoucí bakalářské práce: Ing. Jan Roupec, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2012/13.

V Brně, dne 13. 02. 2013

L.S.

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Děkan

ABSTRAKT

Cílem této bakalářské práce je seznámení se s grafickým vývojovým prostředím Irrlicht engine a následné vyvinutí ukázkové aplikace se zaměřením na Ústav automatizace a informatiky Fakulty strojního inženýrství. V první části se práce věnuje historii a nejdůležitějším součástem engine k tvorbě 3D aplikace. V druhé části je popsána tvorba 3D aplikace s využitím dostupných nástrojů pro tvorbu grafiky.

ABSTRACT

The main goal of this work is to present graphical development enviroment Irrlicht Engine and is develop a sample application focused on promotion of the Instutute of Automation and Computer Science and Faculty of Mechanical Engineering. The first chapter of this work is focused on history and the most important parts of developing 3D applications of Irrlicht Engine. The 3D applications development using available graphic tools is described in the second part of the thesis.

BIBLIOGRAFICKÁ CITACE

BALCÁREK, D. Vývoj aplikací s využitrím grafického vývojového prostředí Irrlicht Engine. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 43 s. Vedoucí bakalářské práce Ing. Jan Roupec, Ph.D..

KLÍČOVÁ SLOVA

Irrlicht engine, 3D engine, 3D herní aplikace, Blender

KEYWORDS

Irrlicht engine, 3D engine, 3D gaming application, Blender

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem práci vypracoval samostatně dle rad a pokynů vedoucího práce pana Ing. Jana Roupce Ph.D. s použitím literárních pramenů a bibliografických citací uvedených v práci.

PODĚKOVÁNÍ

Chtěl bych poděkovat svému vedoucímu práce panu Ing. Janu Roupci Ph.D. za pomoc a rady při tvorbě této bakalářské práce.

Obsah

Slovník pojmů

Engine – počítačový termín, který znamená jádro počítačové hry, databázového stroje nebo programu
Open source (otevřený software) - počítačový software s otevřeným zdrojovým kódem.
DirectX - v informatice sada knihoven poskytujících aplikační rozhraní pro umožnění přímého ovládání moderního hardwaru
OpenGL - průmyslový standard specifikující multiplatformní rozhraní pro tvorbu aplikací počítačové grafiky
OpenGL ES - podmnožinou OpenGL 3D grafického rozhraní určené pro zabudované systémy, jako jsou mobilní telefony
API (Application Programming Interface) – sbírka procedur, metod nebo tříd obsažených v knihovně (programu). Určuje, jakým způsobem se metody z této knihovny volají
Rendering - tvorba reálného obrazu na základě počítačového modelu, nejčastěji 3D
Shader - počítačový program sloužící k řízení jednotlivých částí programovatelného grafického řetězce grafické karty (GPU)
BSP trees - reprezentace objektů v prostoru pomocí stromové struktury dat
FPS - frekvence, s jakou zobrazovací zařízení zobrazuje jednotlivé unikátní snímky. Snímková frekvence se obvykle udává v jednotkách fps (frames per second) nebo v hertzích
Octree - stromová datová struktura, ve které každý vnitřní uzel má právě osm potomků.
GUI - uživatelské rozhraní, které umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků
Triangle fan - jednoduchá technika v 3D počítačové grafice, která zkracuje čas procesu. Popisuje několik spojených trojúhelníků se společným vrcholem
XML parser, writer - programy pro zápis a čtení do XML souborů
Paralax mapping (offset maping, virtual displacement mapping) - metoda aplikovaná na textury v 3D aplikacích
Sphere mapping (spherical environment mapping) - druh zpracování odrazu, který přepočítává odrazovost povrchu a okolní prostředí přitom bere v úvahu jako nekonečně vzdálenou sférickou zeď
Bumb mapping - technika počítačové grafiky sloužící pro simulaci kůže a vrásčitých povrchů
Heightmap – rastrový obrázek sloužící k ukládání hodnot. Například pro údaje o výšce povrchu, které jsou využity u vykreslování terénu
Hardwarová akcelerace - zvýšení funkčnosti počítačového hardware. Například využití výkonu grafické karty a díky tomu současné snížení počtu operací u procesoru
FSAA (Full Screen Anti-Aliasing) - technika vyhlazování hran pro realističtější obraz, kterou používají 3D grafické karty. Využívá dvě základní metody Supersampling a Multisampling
Vertex - uzlový bod nebo vrchol dvou dotýkajících se úseček
Edge - úsečka tvořená body
Face - plocha tvořená úsečkami a body
UV souřadnice - souřadnice v textuře a v daném vrcholu. Textura má v každém směru rozsah souřadnic 0.0 až 1.0, tento rozsah nezávisí na velikosti textury
Alfa kanál - složka pixelu, která udává hodnotu průhlednosti
Morph Target - technika, kde základní uloženou pozici bodů zdeformujeme a při animaci se poloha bodů mění lineárně ze základní polohy na polohu deformovanou[2]
Skeletal - animace spočívá ve vytvoření série navzájem propojených kostí, kterým se následně přiřadí pohyb[3]
Mesh - síť tvořící povrch 3D modelů. Je tvořena vertexy a ploškami

Úvod

Cílem této bakalářské práce je seznámení se s Irrlicht engine a následné vytvoření 3D herní aplikace s využitím tohoto grafického vývojového prostředí.

Grafický engine je vykreslovací jádro grafické aplikace. Umožňuje vykreslování virtuálního prostředí. V tomto prostředí se nachází uživatel, který může provádět jednu z předem definovaných činností. Mezi důležité vlastnosti grafického jádra se řadí rozsah činností, které uživatel může vykonávat a schopnost vykreslit výsledný obraz v co nejkratším čase. Důraz je kladen na zobrazení co největšího počtu snímků za sekundu z důvodu základního požadavku uživatele, dosažení vjemu reálného pohybu.[12]

Irrlicht je multiplatformní grafický engine široce využívaný díky své nezávislosti na platformě a množství podporovaných grafických rozhraní pro programování aplikací. Výhodou tohoto programu je jednoduchost programování a podpora několika programovacích jazyků. V roce 2005 se stal zdrojový kód volně šiřitelný a uživatelé mohou využívat Irrlicht pro jakékoli účely včetně komerčních. Těmito prvky si Irrlicht získal rozsáhlou komunitu uživatelů, kteří přispívají k zdokonalování engine.

První 3D herní aplikace byly vytvořeny v 70. letech 20. století. Tyto aplikace používaly pseudo 3D technologii, spočívající ve vytvoření 2D mapy, která je uživateli s jistými omezeními zobrazována jako 3D. Využívaly jednoduché geometrické tvary a vektorovou grafiku. Plnohodnotná 3D grafika v herních aplikacích začala vznikat v první polovině 80. let, kdy se uživatel mohl pohybovat všemi šesti směry. Grafické zpracování bylo stále jednoduché. Oproti tomu u 2D her se detailnost objektů i grafické zpracování výrazně zlepšila. V roce 1996 byla vytvořena hra Quake, která postavila trojrozměrné hraní na osobních počítačích. Postavy i předměty této hry byly plně trojrozměrné, vymodelované z polygonů. Původní verze Quake podporovala jen softwarový rendering, s příchodem další verze v roce 1997 už podporovala OpenGL rozhraní, což vedlo k rozsáhlejší podpoře grafických akcelerátorů. Od vzniku Quake až do současnosti je kladen stále větší důraz na grafické a dynamické zpracování objektů.[13]

Jelikož Irrlicht je grafický engine, neobsahuje všechny prvky pro vývoj 3D herní aplikace. K vývoji je potřeba využít knihovny, které pracují s fyzikou, se zvukem a v případě potřeby i se sítí, pro aplikace využívající propojení počítačů. Knihoven zabývající se výše uvedenou tématikou je velké množství.

Následující odstavec se věnuje rozčlenění projektu na dílčí cíle a postupu vývoje aplikace. Prvním krokem vývoje je zvolení žánru (akční, strategie, závodní, aj). K tomuto kroku také patří vytyčení cíle, kterého uživatel dosáhne na konci hry. Následně je nutné přejít ke grafickému zpracování aplikace. Grafické zpracování aplikace zahrnuje návrh, modelování, animování objektů a scenérie a také nanášení textur na objekty. Dalším krokem je sjednocení vytvořených objektů do jedné mapy a nastavení kamery neboli výřezu, který umožňuje uživateli prohlížet a pohybovat se ve scenérii. Předposledním krokem je programování aplikace. V poslední části je nutné aplikaci vyexportovat do formátu spustitelného na jakékoliv platformě a vytvořit dokumentaci pro instalaci aplikace.

1 Irrlicht engine

1.1 Popis Irrlichtu

Irrlicht je multiplatformní open source 3D grafický engine, jehož zdrojový kód je volně šiřitelný pod licencí zlib/libpng, která dovoluje používat tento software pro jakékoliv účely, včetně komerčních. Při použití engine není nutné uvádět původ. Pokud je původ uveden nesmí být nějakým způsobem zkreslený. Pokud provedeme změny v engine, musí být jasně označeny a výsledek nesmí být uváděn jako původní software.

Irrlicht engine je nezávislý na platformě, oficiálně pracuje na operačních systémech Windows, Linux, OSX, Sun Solaris / SPARC. Využívá všechny platformy SDL a probíhá vývoj OpenGL ES, které umožní programovat hry pro mobilní telefony. Cílem vývojářů je zjednodušit práci uživateli tak, aby nemusel po vytvoření aplikace změnit jakýkoliv řádek kódu a aplikace fungovala na všech podporovaných platformách.

Engine je napsaný v jazyce C++ a zcela objektově orientovaný. Podporuje i další programovací jazyky, Java, Delphi, C#, Visual Basic. Ve snaze bezproblémového běhu aplikace je engine vybaven několika API rozhraními pro renderování objektů, Direct3D 8.1, Direct3D 9.0, OpenGL 1.2 – 3.x, Irrlicht engine software render, Burningsvideo software render. Rendering se provádí pomocí hierarchického grafu scény. Body scény mohou být například kamera, animovaná postava, animovaná voda. Irrlicht podporuje dvě techniky počítačové animace postav, Morph Target a Skeletal.

Disponuje rozsáhlou knihovnou materiálů pro rychlé vytváření realistických prostředí s podporou vertex a pixel shaderů. Pro tvorbu realistických prostředí je také v engine obsaženo mnoho speciálních funkcí, například pro mlhu, oheň, vodu a jiné. Jejich počet je neustále zvyšován jak tvůrci engine, tak i komunitou.

Primárně je Irrlicht navržen jako grafický engine, neobsahuje knihovny pro práci se zvukem, fyzikou nebo umělou inteligencí. Existuje mnoho různých open source aplikací nebo knihoven, se kterými můžeme interaktivní 3D aplikaci vytvořit.[1]

1.2 Historie Irrlichtu

V roce 2001 vytvořil rakouský programátor Nikolaus Gebhardt demo verzi Irrlicht engine. Už tato verze obsahovala několik zajímavých částí, například práce s BSP stromy nebo animaci postav. O dva roky později byl irrlicht licencován pod lib/libpnq licencí. Od první verze engine bylo vytvořeno dalších 18 verzí, které jsou stručně popsány níže.

- 3.4.2003 – verze 0.1 a 0.1.5 – verze 0.1.5 je oprava verze 0.1 kde byla spousta částí, které nepracovaly správně. Verze pracovaly na operačním systému Windows a obsahovaly tyto hlavní části:
  - rozhraní pro manipulaci s dynamickým osvětlením
  - OpenGL 2D a 3D zařízení
  - ovládání kamery FPS
  - Skeletal animace
  - práce s BSP stromy
  - GUI prostředí
  - podpora modelovacích formátů BSP, OBJ, MD2
  - podpora formátů textur JPG, BMP, TGA, PSD
  - Octree

- 19.5.2003 – verze 0.2
  - podpora 3DS formátů s materiály i texturami
  - skybox
  - billboard
  - opravy metod pracujících s kamerou

- 18.7.2003 – verze 0.3
  - podpora pro Linux
  - podpora MS3D formátů
  - opravy metod pracujících s kamerou
  - úpravy metod pracujících s texturami

- 4.9.2003 – verze 0.4
  - detekce kolizí
  - třídy pro tvorbu ohně, výbuchu, sněhu, dýmu a jiné
  - podpora trilinear filtering
  - message box

- 17.2.2004 – verze 0.5
  - podpora DirectX 9
  - nové materiály
  - mlha
  - nové GUI prostředky
  - podpora kolečka myši
  - update ovládání klávesnice
  - podpora triangle fan

- 19.3.2004 – verze 0.6
  - nové GUI prostředky
  - XML parser a XML writer

- 11.9.2004 – verze 0.7
  - podpora .NET jazyků
  - podpora mipmapping
  - vertex a pixel shadery pro OpenGL a Direct3D8 a 9

- 19.2.2005 – verze 0.8
  - podpora HLSL materiálů
  - podpora BMP formátů

- 28.3.2005 – verze 0.9
  - render terénu
  - podpora OCT, CSM, LMTS, MY3D formátů
  - vykreslování 2D polygonů

- 26.5.2005 – verze 0.10.0
  - parallax mapping
  - podpora referování do textur
  - podpora PNG formátů
  - podpora COLLADA a DMF formátů

- 19.4.2006 – verze 1.0
  - podpora GLSL
  - k dispozici 6 renderů

- 6.9.2006 – verze 1.1
  - import 3DS a OBJ formátů
  - podpora pro B3D a PAK formáty

- 29.11.2006 – verze 1.2
  - podpora hardwarové akcelerace

- 15.3.2007 – verze 1.3
  - v této verzi došlo k výrazným úpravám engine a GUI rozhraní

- 30.11.2007 – verze 1.4
  - v této verzi došlo k výrazným úpravám engine a GUI rozhraní
  - přepsán systém skeletal animace
  - vyvinut fotmát .irrmesh

- 15.12.2008 – verze 1.5
  - FSAA pod OpenGL a Win32
  - ukládání 3D mesh na GPU

- 23.9.2009 – verze 1.6
  - podpora TAR a GZ formátů
  - podpora PLY formátů
  - v této verzi došlo k výrazným úpravám stroje a GUI rozhraní

- 3.2.2010 – verze 1.7
  - v této verzi došlo k výrazným úpravám engine a GUI rozhraní

- 17.11.2012 – verze 1.8 je poslední doposud vydaná verze. Od verze 1.4 jsou změny prováděny pouze s engine, popřípadě se aktualizují podpory formátů nebo ovladačů.[4]

1.3 Popis základních kamenů Irrlichtu

Každý program má své hlavní součásti. Bez znalosti těchto součástí je velmi obtížné tento program ovládat. Irrlicht engine nemá grafické vývojové prostředí, vytváření aplikace probíhá jen pomocí programování. Tato kapitola se zabývá popisem nejnutnějších částí potřebných pro tvorbu jakékoliv aplikace v Irrlichtu.[15]

Jmenné prostory

Hlavní jmenný prostor je irr, ve kterém je zahrnut celý engine. Tento prostor je dále rozdělen na dalších 5 jmenných podprostorů, které jsou popsány níže.

- core – v tomto jmenném podprostoru jsou obsaženy základní třídy jako vektory, pole, seznamy a jiné
- gui – obsahuje třídy pro tvorbu grafického uživatelského rozhraní
- io – poskytuje rozhraní pro vstup a výstup, například čtení a zápis souborů, přístup k archivovaným souborům
- scene - jmenný prostor pro management scény, animace, modelování a jiné
- video – obsahuje přístup k ovladači pro grafické karty, zde se provádí všechny 2D a 3D vizualizace[5]

1.3.1 Nejdůležitější části pro tvorbu aplikace

CreateDevice

Jednou z nejdůležitějších funkcí Irrlicht engine je funkce createDevice, která vytvoří kořenový objekt pro jakoukoliv práci v engine. Funkce má 7 parametrů:

- deviceType – typ zařízení, zde si vybíráme jeden ze softwarových renderů, může nabývat i nulové hodnoty
- windowSize – velikost zobrazovacího okna
- bits – množství bitů na pixel (16 nebo 32)
- fullscreen – boolean – spouštění celé obrazovky
- stencilbuffer – boolean – parametr pro povolení vykreslování stínů
- vsync – boolean - vertikální synchronizace, parametr je použitelný pouze v módu celé obrazovky
- receiver – vytvoří událost, která může být přijata objektem

Smyčka hry

Smyčky for a while jsou velice důležité součásti jakékoliv herní aplikace. Používají se například ke kontrole kolizí, aktualizaci snímku, pro vyhodnocení výhry nebo prohry.

IVideoDriver

Rozhraní pro ovladač, které obsahuje veškeré 2D a 3D grafické funkce. Je jedna z nejdůležitějších částí Irrlicht engine, veškerá manipulace s texturami, vykreslováním je obsažena v tomto rozhraní.

ISceneManager

Zde se vytváří uzly scény, vizualizace terénu, vnitřní vizualizace, tvorba kamer, osvětlení. Umožňuje načítání odlišných formátů modelů, které engine podporuje.

IGUIEnvironment
Tvorba a správa grafického uživatelského rozhraní.

1. Ukázka kódu základní aplikace

Zobrazí objekt ve formátu .obj a načte texturu nanesenou podle UV souřadnic. Pro kompilaci kódu bez chyby je potřeba vložit správnou cestu k objektu a textuře.

```
#include <irrlicht.h>

using namespace irr;
using namespace core;
using namespace video;
using namespace scene;

    int main()
    {   //vytvoření zařízení
      IrrlichtDevice* device =
      createDevice(video::EDT_OPENGL,dimension2d<u32>(640,
      480),16,false,false,false,0);
      //kontrola zařízení
      if(!device)
            return 1;
      //definice ukazatele pro ovladač videa a ISceneManager
      IVideoDriver* driver = device->getVideoDriver();
      ISceneManager* smgr = device->getSceneManager();
      //načtení objektu
      IMesh* mesh = smgr->getMesh("../../media/kostka1.obj");
      //zobrazení objektu ve scéně
      IMeshSceneNode * node = smgr->addMeshSceneNode(mesh);
      //přiřazení světla a textury
      if(node)
      {
            node->setMaterialFlag(EMF_LIGHTING,false);
            node->setMaterialTexture(0,driver-
            >getTexture("../../media/kostka_mapa.png"));
      }
      //nastavení kamery
      smgr->addCameraSceneNode(0,vector3df(20,40,30),vector3df(0,0,0));
      //smyčka zobrazení(vykreslení)
      while (device->run())
      {
            driver->beginScene(true, true, SColor(255, 255, 255,255));
            smgr->drawAll();
            driver->endScene();
      }
      Device->drop();
      return 0;
    }
```

1.3.2 Mesh a podporované formáty

Mesh je seznam vrcholů, hran a ploch, které definují tvar mnohostěnného objektu v 3D počítačové grafice a modelování. Plochy se obvykle skládají z trojúhelníků, čtyřúhelníků a jiných jednoduchých mnohoúhelníků.

Jednou z výhod využití Irrlichtu je jeho komplexnost v použití 3D formátů. V tabulce 1. a 2. jsou stručně popsány formáty, se kterými engine umí pracovat.

Název	Zkratka	Popis
Irrlicht scéna	.irr	Irrlicht dynamický formát schopný pojmout všechny druhy dat.
Irrlicht statická scéna	.irrmesh	Irrlicht formát pro statické objekty.
3D studio meshes	.3ds	Formát používaný 3D modelovacím a animovacím softwarem Autodesk's 3DS Max. Je to binární formát, který obsahuje několik omezení, mesh nesmí obsahovat více než 65 536 trojúhelníků a jména materiálů nesmí být delší než 16 znaků.
AliasWavefront Maya	.obj	Univerzální formát, který může být vytvořen ve většině 3D modelovacích nástrojích. Je to jednoduchý formát nesoucí informaci o mesh a UV souřadnicích textury.
Lightwave objekty	.lwo	Binární formát používaný v 3D modelovacím nástroji LightWave. Formát může obsahovat jeden objekt nebo více propojených objektů popisujících jeden logický objekt a informace o texturách.
Ogre meshes	.mesh	Nativní formát používaný v grafickém engine OGRE. Je to open source formát a je možné jej exportovat z velkého množství modelovacích softwarů. Formát je textový (čitelný), proto méně efektivnější než formáty .obj nebo .3ds.
COLLADA 1.4	.xml, .dae	Formát původně vytvořený společností Sony Computer Entertainment. Formát byl vytvořen za podpory téměř všech atributů 3D modelu. Byl použit v desítkách videoher a 3D modelovacích programech. Formát má velkou budoucnost s možností zahrnutí i fyzikálních atributů do modelu.
My3DTools	.my3D	Formát umožňující přímý export lightmap scény z 3DS Max do Irrlicht engine
Pulsar LMTools	.lmts	Formát umožňující export lightmap scény z 3DS Max.
Quake 3 levels	.bsp	Formát byl původně vytvořen pro Doom engine. Je to archivovaný formát, který používá algoritmus binárního dělení prostoru, minimalizující čas načítání a vykreslování 3D objektů.
DeleD	.dmf	Nativní formát pro open source software deleD, který je určen pro tvorbu grafického prostředí v počítačových hrách a jednoduché 3D grafiky.

FSRad oct	.oct	Souborový formát obsahující informace o textuře a vypočtených lighmaps.
Cartography shop 4	.csm	Formát, který používá open source software pro tvorbu herních úrovní.
STL 3D	.stl	Byl původně navržen pro CAD data, proto má špatnou podporu pro barvy a textury.
PLY 3D files	.ply	Souborový formát známý jako Polygon File Format navržen podle formátu .obj a obsahuje několik jedinečných funkcí, například přiřazení různých vlastností na dvou různých stranách.

Tabulka 1 – přehled formátů, které podporuje Irrlicht engine[11]

Název	**Zkratka**	**Popis**	**Skeletal animace**	**Morph Target animace**
B3D Files	.b3d	Formát vytvořený společností Blitz Basic.	Ano	Ne
Microsoft DirectX	.x	Microsoft 3D formát vytvořený pro Direct X SDK. Formát je k dispozici v binární i textové formě.	Ano	Ne
Milkshape	.ms3D	Milkshape animační formát.	Ano	Ne
Quake 2 models	.md2	Velmi efektivní animační soubory.	Ne	Ano
Quake 3 models	.md3	Vylepšený formát md2.	Ne	Ano

Tabulka 2 – přehled animačních formátů, které podporuje Irrlicht engine[11]

2. Ukázka části kódu aplikace

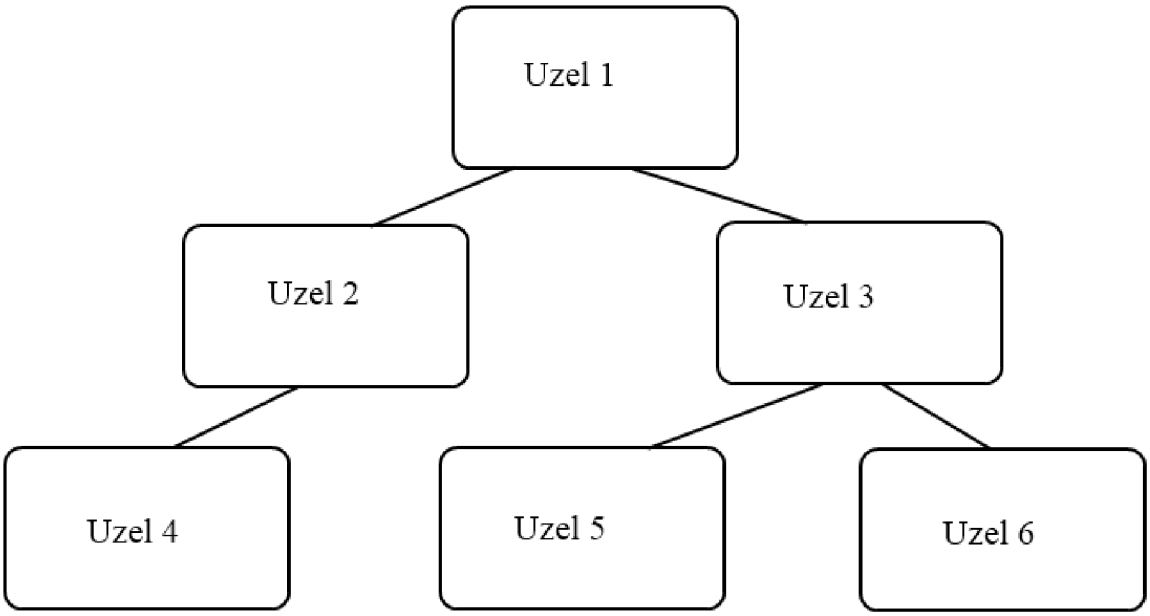
Ukázka zobrazuje vytvoření ukazatelů na třídu ISceneManager, která spravuje meshes, uzly, camery a jiné objekty v Irrlicht engine. Dále vytvoří ukazatel pro mesh, který chceme načíst. Nakonec zkontroluje, zda je v ukazateli. Pokud ne ukončí aplikaci.

```
//tvorba ukazatele ke třídě ISceneManager
ISceneManager* smgr = device->getSceneManager();
//načtení objektu
IMesh* mesh = smgr->getMesh("cesta k objektu");
//kontrola načtení objektu
if(!mesh)
{     device->drop();
      return 1;
}
```

1.3.3 Uzly a kamery

Uzly

Uzly jsou základní jednotkou využívanou k vytváření souvisejících datových struktur, jako jsou stromové a seznamové struktury. Každý uzel obsahuje data a možná propojení s dalšími uzly. Odkazy mezi uzly jsou realizovány pomocí ukazatelů nebo referencí.



Obrázek 1- členění uzlů do struktury

Uzel 1 je hlavní nebo kořenový uzel. Uzly 2 a 3 jsou vzhledem k uzlu 1 označovány jako potomci. Vzhledem k tomu, že uzly 2,3 mají předka i potomky, nazývají se vnitřními uzly. Uzel 4 je potomek uzlu 2 a uzly 5,6 jsou potomky uzlu 3. Uzly 4,5,6 nemají žádného potomka, nazývají se tedy konečnými uzly.

Každý objekt, který je vykreslen v Irrlicht engine je uzel. Pokud chceme zobrazit mesh, který jsme načetli, musíme vytvořit uzel. Ke všem uzlům se přistupuje pomocí třídy ISceneManager.

3. Ukázka části kódu aplikace

Vytvoříme ukazatel na třídu IMeshSceneNode, díky kterému zobrazíme statický mesh. Dále vypneme dynamická světla, aby objekt nebyl černý. Nakonec nastavíme texturu, která náleží na objekt.

```
//zobrazení objektu
IMeshSceneNode * uzel = smgr->addMeshSceneNode(mesh);
//jestlize máme uzel
if(uzel)
{     //vypneme světla
      node->setMaterialFlag(EMF_LIGHTING,false);
      //nastavíme texturu
      node->setMaterialTexture(0,driver->getTexture("cesta k
      textuře"));
}
```

Kamera

Je výřez, který umožňuje uživatelům prohlížet scénu, která je vykreslena na obrazovce. V Irrlicht engine jsou 3 typy kamer, které může uživatel využít. Normální kamera, FPS kamera a Maya kamera.

Kamera	Popis
Normální	Nereaguje na vstup uživatele. Pokud chceme ovládat směr, musíme zavolat příslušnou metodu.
FPS	Zajišťuje ovládání myši a klávesnice a je vhodná pro hry z pohledu první osoby (akční).
Maya	Standardní kamera poskytující ovládání pomocí myši. Pokud se kamera vztahuje k nějakému objektu, je potřeba nastavit příslušnou vzdálenost od objektu.

Tabulka 3 – Stručný popis podporovaných kamer

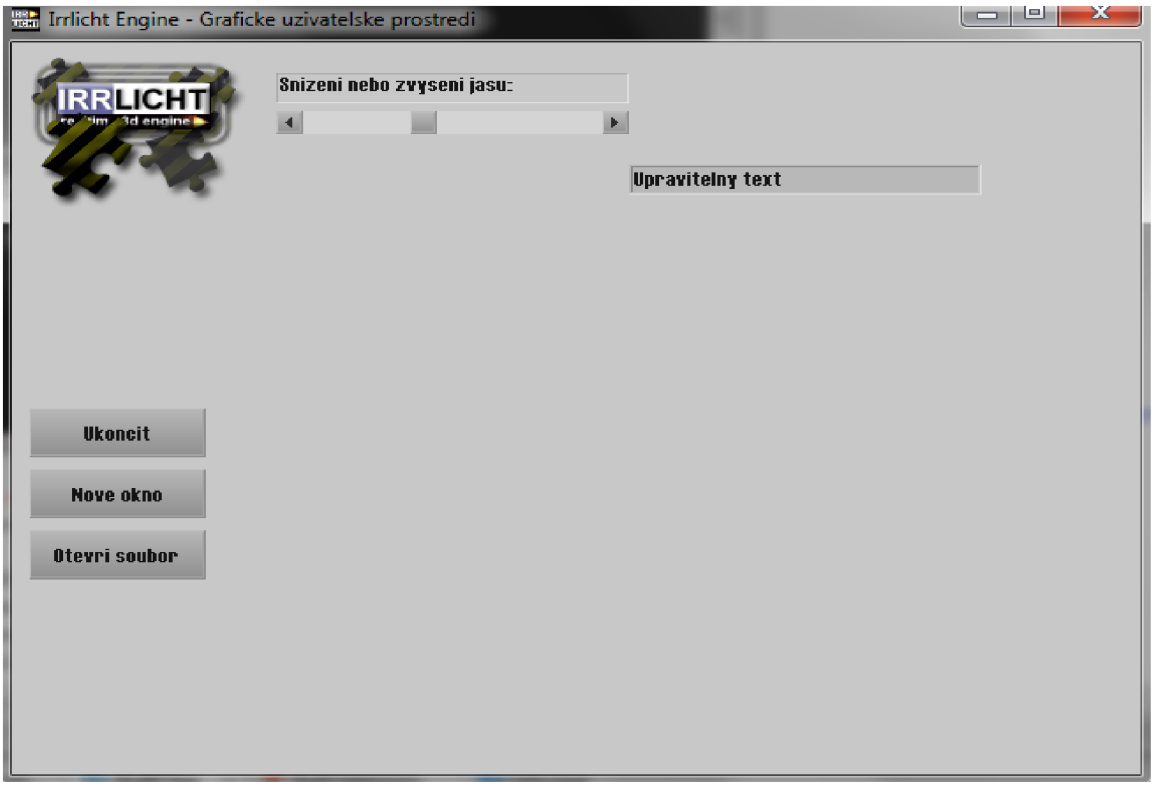
4. Ukázka části kódu aplikace

Vytvoření kamery. První vektor udává pozici kamery a druhý natočení kamery.

```
//Vytvoření kamery
smgr->addCameraSceneNode(0,vector3df(0,0,0),vector3df(0,0,0));
```

1.3.4 Grafické uživatelské prostředí

Irrlicht engine má vestavěné grafické uživatelské rozhraní, které slouží k ovládání počítače pomocí interaktivních grafických ovládacích prvků. V Irrlichtu je možné vytvořit například tyto prvky: okna, tlačítka, posuvníky, statický text a seznam výběru.



Obrázek 2 – ukázka grafického uživatelského prostředí

IEventReciever

Pro použití grafického uživatelského prostředí nebo pohybu s objektem pomocí vstupních zařízení se v Irrlicht engine používá třída IEventReciever. Tato třída slouží ke komunikaci se vstupními zařízeními například myš, klávesnice a jiné. Princip spočívá v kontrole stisku tlačítek a přiřazením vlastností objektu, který má vykonávat nějakou činnost.

5. Ukázka části kódu aplikace

Ukázka zobrazuje vytvoření okna pro vkládání textu. První vytvoříme ukazatele na třídu IGUIEnvironment a následně vložíme pomocí funkce addEditBox() okno. Prvním parametrem funkce je text v okně a druhý je tvar a velikost.

```
//ukazatel na uživatelské grafické rozhraní
IGUIEnvironment* env = device->getGUIEnvironment();
//přidání EditBoxu
env->addEditBox(L"Upravitelny text", rect<s32>(350, 80, 550, 100));
```

1.3.5 Kolize

Detekce kolizí se provádí pomocí třídy ITriangleSelector, která shromažďuje trojúhelníky na zobrazovaném objektu. Tato třída obsahuje hned několik selektorů. Odlišnost spočívá v objektu, který používáme, například pro animovaný a statický objekt nebo vykreslený terén.

6. Ukázka části kódu aplikace

Vytvoříme ukazatel na třídu ITriangleSelector. Dále vytvoříme OctTreeTriangleSel -ector, který je vhodný pro velké množství trojúhelníků. Přiřadíme ho danému uzlu a zahodíme ho.

```
//ukazatel
ITriangleSelector * selector = 0;
//vytvoření selektoru na potřebný mesh a uzel
selector = smgr->createOctTreeTriangleSelector(mesh, meshUzel);
//přiřadíme ho uzlu
meshUzel->setTriangleSelector(selektor);
//zahodíme
selektor->drop();
```

1.3.6 Speciální funkce

Irrlicht disponuje spoustou speciálních efektů, které jsou neustále doplňovány programátory nebo komunitou. V poslední vydané verzi 1.8 jsou k dispozici tyto efekty:

- Animované vodní plochy
- Dynamická světla
- Dynamické stíny
- Animace textur
- Mlha
- Obloha (Skybox)
- Průhledné objekty
- Parallax mapping
- Bump mapping
- Billboardy
- Upravitelné částicové efekty pro vytváření efektu sněhu, kouře, ohně
- Sphere mapping

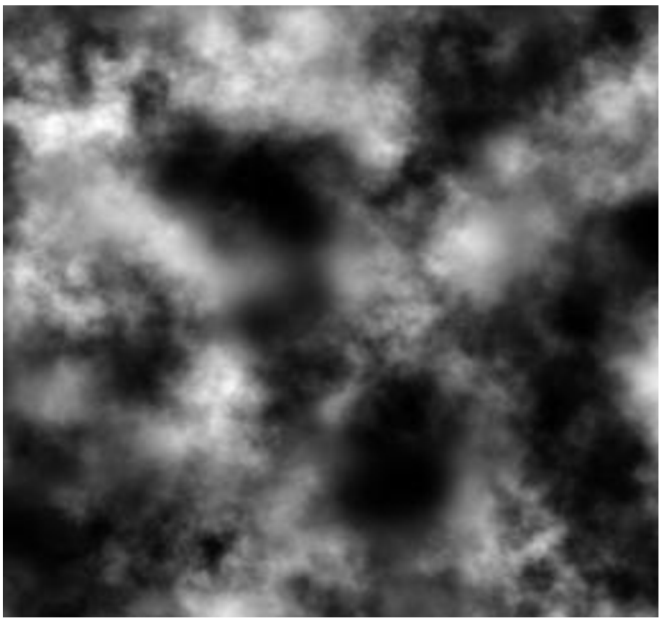
7. Ukázka části kódu aplikace

Pokud chceme využívat dynamické stíny, musíme ve funkci createDevice() nastavit parametr pro použití stínů na hodnotu true. Pomocí uzlu scény vytvoříme stín jako potomka tohoto uzlu. Dále ukazatelem na třídu ISceneManager přiřadíme barvu stínu.

```
uzel->addShadowVolumeSceneNode();
smgr->setShadowColor(video::SColor(100,0,0,0));
```

1.3.7 Vykreslování terénu

Vykreslování terénu je řešeno pomocí heightmapy, která je na terén nanesena. Tato mapa, je černobílý rastrový obrázek sloužící k ukládání dat, například dat terénu. Metoda spočívá ve vyvýšení světlých částí a snížení tmavých. Jednoduchá a velmi rychlá metoda pro vykreslování terénu.



Obrázek 3- ukázka heighmapy[14]

8. Ukázka části kódu aplikace

Vytvoříme ukazatel na třídu ITerrainSceneNode, kterému předáme parametr cesty k textuře heightmapy. Dále nastavíme 2 textury a použijeme materiál EMT_DETAIL_MAP. Tento materiál se používá k vykreslování terénu. Nakonec nastavíme velikost textur, kdy s nastavenými parametry pokryje terén první textura 1x a druhá textura 20x. To způsobí detailnější terén s různými odstíny.

```
//ukazatel
ITerrainSceneNode* teren = smgr->addTerrainSceneNode("cesta
k heightmapě");
//textury
teren->setMaterialTexture(0,driver->getTexture("cesta k
textuře"));
teren->setMaterialTexture(1,driver->getTexture("cesta k
textuře"));
//materiál
teren->setMaterialType(video::EMT_DETAIL_MAP);
//velikost textur
teren->setScaleTexture(1.f,20.f);
```

2 Tvorba aplikace

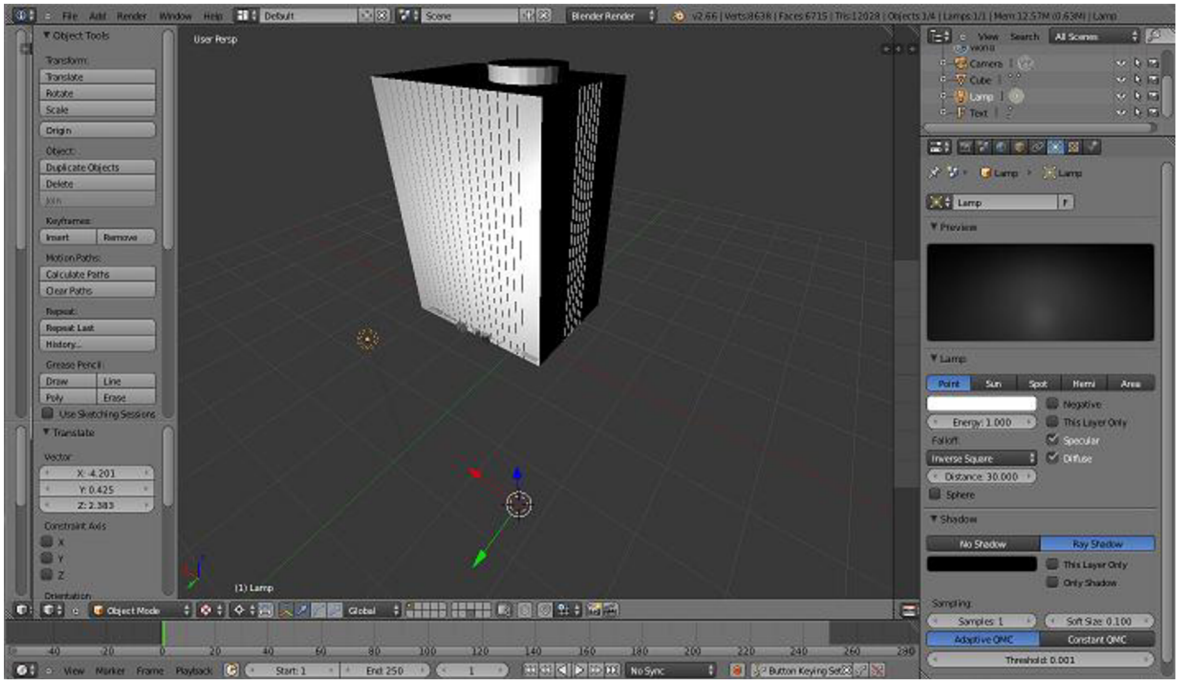
Vývoj herní aplikace se skládá ze tří důležitých částí. První a nejdůležitější částí je scénář, děj nebo cíl, který povede hráče celou aplikací a dokáže udržet jeho pozornost na co nejdelší dobu. Druhou důležitou částí je hratelnost, ve které je obsaženo ovládání, fyzika, nastavení kamery a jiné prvky, které slouží ke komunikaci mezi uživatelem a aplikací. Poslední částí je grafika herní aplikace. Při tvorbě grafiky je kladen důraz na dosažení co největší realističnosti.

Smysl této ukázkové aplikace spočívá v závodní hře pro dva hráče. Uživatel se snaží dojet do cíle v co nejkratším čase. Závodní dráha je volně navržena podle areálu VUT v Brně, fakulty strojního inženýrství.

2.1 Modelování

Při modelování byl kladen důraz na rozumnou detailnost objektů, tak aby grafické zpracování ukázky bylo jednoduché a hratelné na jakémkoliv počítači. Obecně platí, čím méně vertexů, tím lépe. Jako modelovací nástroj byl zvolen Blender, který je multiplatformní open source program zaměřený na tvorbu 3D modelů, animací, renderingu a jiných. Díky hernímu engine je možné taky vytvářet jednoduché interaktivní aplikace.[6]

Většina objektů byla modelována ze základních geometrických tvarů, pomocí funkcí programu Blender. Tvorba aut spočívala v technice obrázku na pozadí, kdy podle obrysu objektu v zobrazovaném obrázku byl vytvořen 2D model a následně pomocí Blender funkcí převeden do 3D. Důležitá funkce, která usnadnila modelování auta je zrcadlo. Funkce nastavuje pozice vertexů podle středové roviny. Rostliny byly vytvořeny jako plochy, na kterých je nanesena textura obsahující alfa kanál. Kvalita zpracování rostlin je nízká, jelikož u nich není nutné zřetelně rozpoznat třetí rozměr.



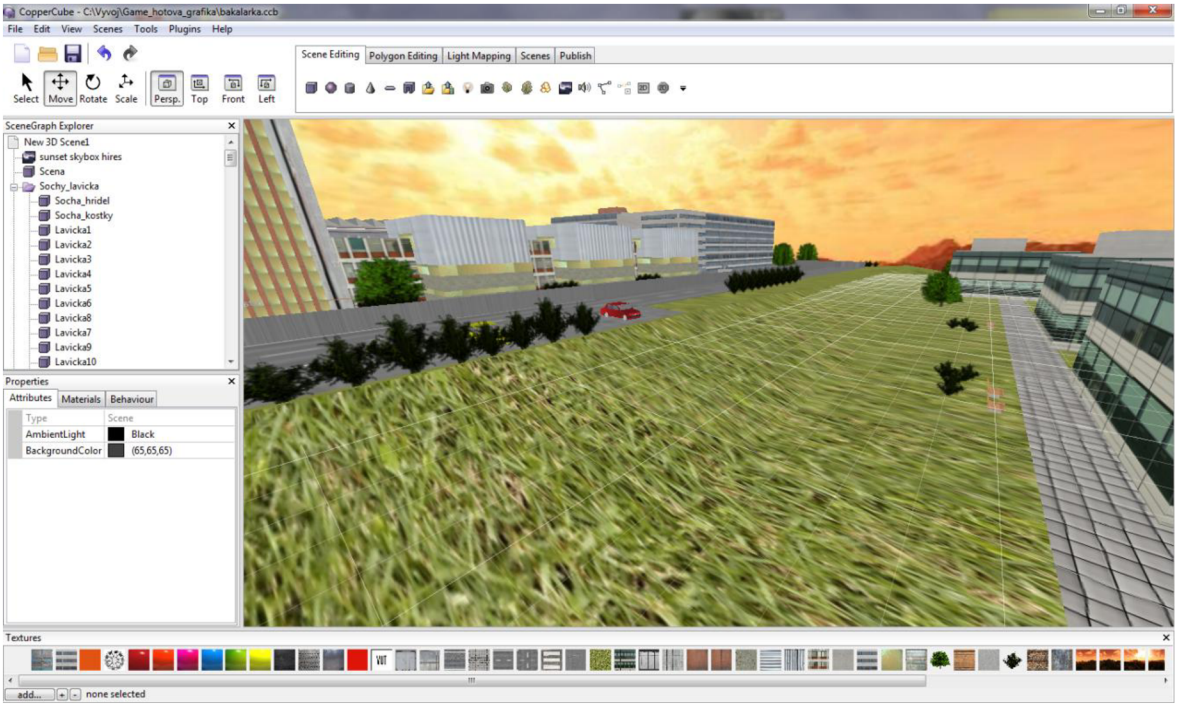
Obrázek 4 – Ukázka vývojového prostředí Blender

2.2 Texturování

Použité textury byly staženy z webových serverů, které obsahují volně přístupné materiály k tvorbě 2D a 3D počítačové grafiky, modelů a aplikací například CGtextures[7], Free 3D texture galery[8], Got3D free textures[9]. Každá textura byla před použitím na samotný objekt upravena v open source multiplatformním grafickém editoru GIMP[10]. Úprava spočívala ve využití základních funkcí pro modifikaci fotografií. GIMP patří mezi často využívaný nástroj pro úpravu fotografií a jeho funkčnost je srovnatelná s drahými programy zabývající se touto problematikou. Po upravení byly textury naneseny v programu Blender pomocí UV souřadnic.

2.3 Tvorba scény

K tvorbě scény byl použit úrovňový editor IrrEdit. Tento 3D úrovňový editor je zdarma a napsán pro společné použití s Irrlicht engine. Vymodelované objekty v Blenderu byly vyexportovány do formátu .obj, který IrrEdit podporuje. Tento formát byl zvolen z důvodu jeho jednoduchosti a rychlosti, pokud by byl některý z objektů animovaný, bylo by nutné použít formát souboru podporující animaci. Scéna byla vytvořena pomocí úpravy pozice a velikosti jednotlivých vymodelovaných objektů. Následně bylo nutné scénu vyexportovat do formátu, který podporuje Irrlicht engine. Pro export byl zvolen formát .irr. Tento formát je XML soubor nesoucí informace o všech použitých objektech.[17]



Obrázek 5 – Ukázka prostředí úrovňového editoru IrrEdit

2.4 Programování aplikace

Tato kapitola se zabývá programováním závodní hry. Pro vytvoření této aplikace byly použity Irrlicht knihovny verze 1.7.2. Samotné programování bylo rozčleněno do několika kroků, načtení scény, nastavení kolizních vlastností, ovládání, nastavení kamery, smyčka aplikace a tvorba grafického uživatelského prostředí. Scéna byla načtena pomocí třídy `loadScene` z .irr formátu, který byl vyexportován úrovňovým editorem IrrEdit. Aby nebylo možné procházet přes objekty, byly nastaveny kolizní vlastnosti a to pomocí smyčky `for` a třídy `ITriangleSelector`. K ovládání objektů byla použita třída `IEventReciever`. Při ovládání objektu, v našem případě auta, se pohled uživatele musí měnit s pozicí objektu. Pozice kamery bylo tedy nutné měnit podle pozice auta. Předposledním krokem byla smyčka aplikace, ve které byly řešeny funkce potřebné k závodní hře. Například odpočet času, dojetí do cíle. Pro ukázku bylo také vytvořeno jednoduché grafické uživatelské prostředí.

U některých kroků je zobrazena ukázka kódu pro představu.

2.4.1 Načtení scény

Po vytvoření scény v úrovňovém editoru IrrEdit, byl pro vyexportování zvolen formát .irr. Tento souborový formát je XML soubor. IrrEdit vytvoří soubor .irr nesoucí hlavní informace o použitých objektech a následně vytvoří složku .irr.meshes. V této složce jsou veškeré objekty použité ve scéně rozdělené do jednotlivých souborových formátů .irrmesh, každý objekt má svůj vlastní XML soubor .irrmesh. V těchto souborech jsou detailní informace o objektu. Formát .irrmesh tedy slouží pro jednotlivé objekty a formát .irr pro scénu.

Načtení scény bylo provedeno pomocí třídy `loadScene`, která načte scénu do třídy `ISceneManager` starající se o vykreslování.

```
ISceneManager* smgr = device->getSceneManager();
//nacitani sceny
smgr -> loadScene("cesta k XML souboru scény");
```

Po úspěšné kompilaci a spuštění skriptu nastal problém s načítáním objektů a textur. Parametr cesty, který byl zadán ve třídě `loadScene` byl správný. Dalším krokem bylo tedy otestovat správnost cest v XML souboru pro načtení scény (formát .irr). Cesta k objektům a texturám byla špatná, byla upravena a následně spuštěná aplikace pracovala správně.

Dalším krokem pro zobrazení scény bylo načtení závodních aut, neboli objektů, se kterými budou hráči pohybovat. Byla vytvořena jednoduchá metoda, která vytvořila uzly načtených objektů a nastavila jim základní vlastnosti jako velikost, vypnula světla a nastavila rotaci.

2.4.2 Nastavení kolizních vlastností

Scéna obsahuje velké množství objektů. Nastavovat kolizní vlastnosti jednotlivě pro každý objekt by bylo zdlouhavé. Byla tedy použita smyčka `for`. Jako parametr pro ukončení této smyčky bylo využito pole. Do tohoto pole byly pomocí metody `ISceneNodeFromType` vloženy všechny objekty ze scény. Do této metody byly vloženy dva parametry, první obsahoval informaci o hledaných objektech a druhý kam je budeme vkládat.

Pro načtení všech objektů ze scény byla použita hodnota `ESNT_ANY` z výčtového typu `ESCENE_NODE_TYPE` obsahující všechny typy uzlů využívaných Irrlicht engine.

```
array<ISceneNode *> uzly;
//vyhledani vsech uzlu a jejich vlozeni do pole
smgr->getSceneNodesFromType(ESNT_ANY, uzly);
//smycka for s ukoncujicim parametrem velikosti pole
for (u32 i=0; i < uzly.size(); ++i)
```

Uvnitř smyčky `for` byly přiřazeny kolizní vlastnosti objektům. Každému objektu z pole byl nastaven selektor. Jak již bylo výše uvedeno, Irrlicht obsahuje několik typů selektorů, každý vhodný pro daný typ objektu. Objekty byly rozlišeny pomocí větvení klíčového slova `switch`. Jako podmínka byla vložena metoda `getType` pro uzly, která vrací hodnotu typu uzlu. Pak už stačilo typ objektu porovnat s výčtem typů `SCENE_NODE_TYPE` pomocí `case`. Za příkazem `case`, byl nastaven danému typu objektu příslušný selektor. Nakonec smyčky musíme tyto selektory vložit do třídy, která shromažďuje selektory tak, aby fungovaly jako jeden. Tato třída se nazývá `IMetaTrinagleSelector` a bylo nutné na ni vytvořit ukazatel před smyčkou `for`.[16]

```
ISceneNode * uzel = uzly[i];
//ukazatel na ITriangleSelector
ITriangleSelector * selector = 0;

//pouziti switch pro porovnani typu objektu
switch(uzel->getType())
   {
   case ESNT_CUBE:
   case ESNT_ANIMATED_MESH:
   //prirazeni kolize animovanemu objektu
   selector = smgr->createTriangleSelectorFromBoundingBox(uzel);
   break;
   //nasleduji dalsi case
   }//konec switch

if(selector)
   {
    //vlozime selektor do do meta selektoru
    meta->addTriangleSelector(selector);
    // Zahodime selector
    selector->drop();
   }
```

Pro ovládaná auta byly kolizní vlastnosti nastavovány pomocí animátoru Response Collision Animator, který provádí detekci kolizí a také příslušné reakce uzlu ke, kterému je přiřazen. V tomto případě byl tento animátor přiřazen ovládaným autům. Auta se nebudou moci pohybovat přes budovy a budou mít nastaveny gravitaci.

2.4.3 Ovládání

Ovládání bylo vyřešeno pomocí třídy `IEventReciever`. Kdy byla vytvořená nová třída `MyEventReciever` jako potomek třídy `IEventReciever`. Aby bylo možné zjistit, zda bylo nějaké tlačítko stisknuto, byla využita v naší nové třídě metoda `OnEvent`, kterou obsahuje třída `IEventReciever`. Pak už bylo potřeba kontrolovat, které tlačítko bylo stisknuto a zapamatovat si aktuální stavy tlačítek.

Samotné ovládání aut spočívalo v kontrole stisknutých tlačítek ve smyčce `while`. Pomocí podmínky `if` a metody vytvořené pro kontrolu stisku tlačítka, která byla volána v podmínce, bylo kontrolováno stisknutí daného tlačítka. Po podmínce následoval kód pro posun a rotaci auta. Proměnné `uzelPozice` a `uzelRotace` jsou 3D vektory sloužící pro posun a zatáčení auta. Do těchto proměnných byly při inicializaci nastaveny hodnoty pozice a rotace ovládaného auta. Tyto pozice byly získány pomocí metod `getPosition` a `getRotation`.

```
//inicializace 3D vektoru pozice a rotace
vector3df uzelPozice = autoUzel ->getPosition();
vector3df uzelRotace = autoUzel ->getRotation();

//kod pro pohyb auta
if(receiver.IsKeyDown(irr::KEY_UP))
//upravuj pozici uzlu auta pomoci vektoru, pohyb vpred
uzelPozice -=MOVEMENT_SPEED * frameDeltaTime * forward1;
else if(receiver.IsKeyDown(irr::KEY_DOWN))
//upravuj pozici uzlu auta pomoci vektoru, pohyb vzad
uzelPozice += MOVEMENT_SPEED * frameDeltaTime * forward1;
if(receiver.IsKeyDown(irr::KEY_RIGHT))
//upravuj rotaci uzlu auta pomoci vektoru, pohyb vpravo
autoUzel->setRotation(uzelRotace + core::vector3df(0, +40 *
frameDeltaTime, 0));
else if(receiver.IsKeyDown(irr::KEY_LEFT))
//upravuj rotaci uzlu auta pomoci vektoru, pohyb vlevo
autoUzel->setRotation(uzelRotace + core::vector3df(0, -40 *
frameDeltaTime, 0));
```

2.4.4 Nastavení kamery

V ukázkové aplikaci byly použity dvě normální Irrlicht kamery. Každá pro jedno z ovládaných aut. Při inicializaci těchto kamer nastavíme jejich pozici pomocí 3D vektorů, které obsahují pozice aut. Jednu ze souřadnic těchto vektorů bylo nutné zvětšit tak, aby bylo v pohledu kamery vidět auto i okolí. Následně ve smyčce `while`, kde nastavujeme pozici a rotaci auta musíme při pohybu auta nastavovat i kameru souběžně s pozicí auta.

```
//vektor pro pozici kamery
vector3df kameraPozice = autoUzel->getPosition();
//nastaveni pozice kamery
kamera->setPosition(vector3df(kameraPozice.X, kameraPozice.Y+4,
kameraPozice.Z));
```

Ukázková aplikace je závodní hra pro dva hráče. To znamená, že každý hráč musí mít svou část obrazovky zobrazující jeho auto i okolí. Obrazovka byla rozdělena na dvě poloviny, každá pro jednoho hráče. Rozdělení bylo realizováno pomocí metody `setViewPort`.

```
//nastaveni aktivni kamery
smgr->setActiveCamera(camera[0]);
//nastaveni vyrezu, který kamerou uvidime
driver->setViewPort(rect<s32>(0,0,ResX/2,ResY));
//vykresleni
smgr->drawAll();
```

2.4.5 Smyčka aplikace

Výsledná aplikace obsahuje 3 smyčky `while`. Každá smyčka se stará o jednu část aplikace. První částí je start. Tato část obsahuje odpočet a nastavení základní pozice kamer. Druhou částí je průběh, kde jsou kontrolovány stisky tlačítek, upravovány pozice kamer, rozdělována obrazovka a obsažena kontrola dojetí do cíle. Třetí částí je konec, kde je vyhodnocena výhra.

Jak již bylo uvedeno výše, v první smyčce `while` je obsažena část Start. Tato část obsahuje jednoduchý odpočet a nastavení pozice kamer. Odpočet byl realizován tak, že ještě před danou smyčkou `while` byla deklarována proměnná `pocatekCas` a inicializována na hodnotu systémového času, který byl zjištěn pomocí metody `getRealTime`. Tato metoda vrací systémový čas v milisekundách. Dále byla ve smyčce `while` deklarována proměnná `aktualniCas` s hodnotou systémového času. Následně byla proměnná `pocatekCas` odečtena od proměnné `aktualniCas` a výsledek přiřazen proměnné `odpocetCas`. Pak už jen stačilo kontrolovat proměnou `odpocetCas`. Pomocí podmínek `if` byla kontrována velikost proměnné v milisekundách a v každé podmínce vypisována potřebná data na obrazovku.

Druhá smyčka `while` se zabývá průběhem. Průběh aplikace je v podstatě závod, kde uživatelé ovládají svá auta, probíhá zde nastavení kamery a rozdělení obrazovky. Ukázky zdrojových kódů a popis těchto částí je uveden výše. V této smyčce je také obsažen kód vyhodnocující konec závodu. Nejprve bylo nutné získat rámy aut. Rámy jsou v tomto případě chápány jako 3D obrys auta. S rámy je následně možné vyhodnotit konec závodu. Tyto rámy získáme pomocí metody `getTransformedBoundingBox`. Pro vyhodnocení závodu byla použita bool metoda `intersectsWithBox`, vracející hodnotu true pokud dojde k průniku dvou rámů.

```
//Podmínka if, ve které zjistujeme prunik
if(autoUzel->
getTransformedBoundingBox().intersectsWithBox(cilUzel->
getTransformedBoundingBox()))
)
{  //pokud ano, vyskocime ze smycky
   break;
}
```

Poslední smyčkou `while` je konec, ve které bylo vyhodnoceno který hráč vyhrál a následně je aplikace ukončena. Při vyhodnocení je použita stejná podmínka jako v předešlé smyčce. V podmínce je už jen vypsán text, který z hráčů vyhrál.

2.4.6 Grafické uživatelské rozhraní

K ukázkové aplikaci bylo vytvořeno i jednoduché grafické uživatelské rozhraní. Pro ukázku obsahuje toto rozhraní dvě tlačítka pro start a konec hry. Dva obrázky jeden s logem Irrlicht engine a druhý s logem fakulty strojního inženýrství a dvě textová okna.

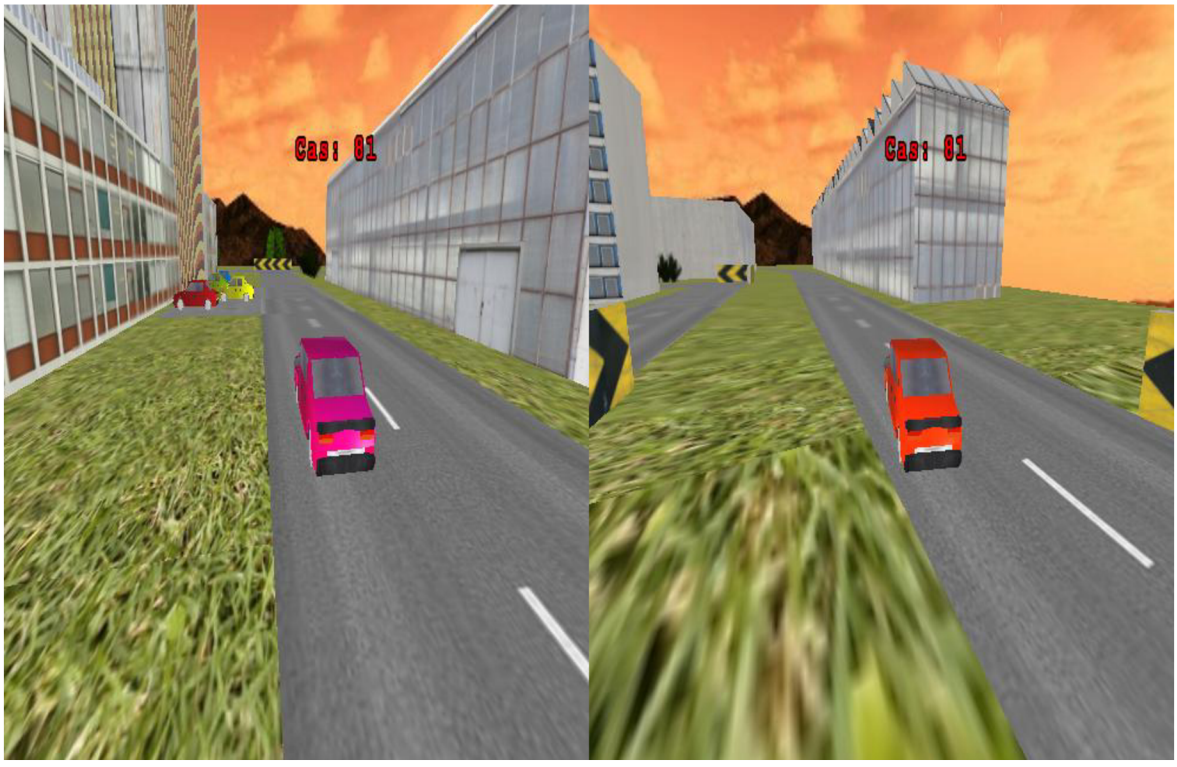
Obrázek 6 - ukázka vytvořeného uživatelského prostředí

2.4.7 Ukázky aplikace

Obrázek 7- ukázka aplikace

Obrázek 8 - ukázka aplikace

Obrázek 9- ukázka aplikace



Obrázek 10- ukázka aplikace

3 Závěr

Cílem této práce bylo seznámení se s grafickým vývojovým prostředím Irrlicht engine a následné vyvinutí ukázkové aplikace, tematicky zaměřené k propagaci studia na ústavu aplikované informatiky.

V první části bylo popsáno grafické vývojové prostředí Irrlicht engine. Historie engine byla uvedena v bodech, ve kterých jsou s příchody nových verzí uvedeny nové části. Dále byly v této části popsány základní kameny Irrlicht engine, nejdůležitější metody, třídy a jmenné prostory pro vývoj jakékoliv aplikace s praktickou ukázkou zdrojového kódu.

V druhé části byla vytvořena ukázková aplikace, závodní hra pro dva hráče. Závodní dráha byla navržena volně podle areálu VUT, fakulty strojního inženýrství.

Vývoj aplikace byl rozdělen do několika dílčích kroků. Návrh objektů a scenérie, kde byla navržena závodní dráha. Druhým krokem bylo modelování a nanášení textur. V tomto kroku byly navržené objekty vymodelovány v 3D modelovacím nástroji. V tomto modelovacím nástroji byly také naneseny textury. Textury byly staženy z webových serverů zabývajících se touto tématikou a upraveny v 2D editoru. Třetím krokem vývoje byla tvorba scény, kde byly veškeré vymodelované objekty sjednoceny do jedné mapy. Posledním etapou vývoje aplikace bylo programování.

Aplikace byla úspěšně dokončena a je možné ji dále rozvíjet s využitím fyzikálních, audio nebo síťových knihoven. Vzhledem k úspěšnému dokončení projektu se tato práce může stát oporou pro tvorbu 3D grafické aplikace v Irrlicht engine.

4 Seznam použité literatury

[1] Features. *Irrlicht Engine - A free open source 3D engine* [online]. 2002 [cit. 2013-03-19]. Dostupné z: http://irrlicht.sourceforge.net/features/

[2] Morph target animation. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2008- [cit. 2013-03-21]. Dostupné z: http://en.wikipedia.org/wiki/Morph_target_animation

[3] Skeletal animation. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2007- [cit. 2013-03-21]. Dostupné z: http://en.wikipedia.org/wiki/Skeletal_animation

[4] GEBHARDT, Nikolaus. *Irrlicht engine* [online]. 2003, 7.11.2012 [cit. 2013-03-21]. Dostupné z: http://irrlicht.sourceforge.net/changes.txt

[5] GEBHARDT, Nikolaus. *Irrlicht Engine 1.8 API documentation* [online]. 2003 [cit. 2013-03-21]. Dostupné z: http://irrlicht.sourceforge.net/docu/index.html

[6] *Blender* [online]. Dostupný z WWW: http://www.blender.org/

[7] VIJFWINKEL, Marcel. *[CG Textures] - Textures for 3D, graphic design* [online]. neuvedeno, 2013-04-10 [cit. 2013-04-10]. Dostupné z: http://www.cgtextures.com/

[8] *Free 3d Texture Gallery* [online]. 2007, 2013-04-10 [cit. 2013-04-10]. Dostupné z: www.3dtexture.net

[9] *Got3d - Free Textures* [online]. 2004, 2013-04-10 [cit. 2013-04-10]. Dostupné z: http://free-textures.got3d.com

[10] *GIMP* [online]. 2001 [cit. 2013-03-17]. Dostupné z: http://www.gimp.org/

[11] *Irrlicht Engine wiki* [online]. 2005 [cit. 2013-04-10]. Dostupné z: http://irrlicht3d.org/wiki/index.php?n=Main.IrrlichtEngineWiki

[12] SVĚT HARDWARE. Grafické enginy her a reálný svět. [online]. Pavel KOVÁČ. [2012] [cit. 2013-04-18]. Dostupné z: http://www.svethardware.cz/graficke-enginy-her-a-realny-svet/18297

[13] PCTUNING. Vývoj technologií počítačových her. [online]. Tomáš ŠULC. [2011] [cit. 2013-04-18]. Dostupné z: http://pctuning.tyden.cz/multimedia/hry-a-zabava/21804-vyvoj-technologii-pocitacovych-her-prvni-dil

[14] Heightmap. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Heightmap

[15] GEBHARDT, Nikolaus. Irrlicht Engine [online]. 2003 [cit. 2013-05-02]. Dostupný z WWW: http://irrlicht.sourceforge.net/

[16] GEBHART, Nicolaus. Irrlicht engine. *Related Pages* [online]. 2003 [cit. 2013-05-08]. Dostupné z: http://irrlicht.sourceforge.net/docu/pages.html

[17] CopperCube - a 3D editor for WebGL and Flash, Mac OS X, Windows and mobile apps. *Ambiera e.U. Software Development* [online]. Neuvedeno [cit. 2013-05-19]. Dostupné z: http://www.ambiera.com/coppercube/index.html

5 Seznam příloh

Obsah CD

- Elektronická verze práce
- Irrlicht verze 1.7.2
- Zdrojový kód aplikace
- Vymodelované objekty a textury
- Manuál pro spouštění aplikace