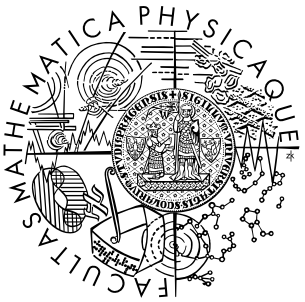
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Ondřej Šindelář

Simulátor motokrosu

Katedra aplikované matematiky

Vedoucí bakalářské práce: Mgr. Tomáš Bílý
Studijní program: Informatika, programování

2007

Chtěl bych poděkovat Balázsovi Rózsovi, autorovi hry Elasto Mania, za inspiraci při vytváření této bakalářské práce a Mgr. Tomášovi Bílému za její vedení.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 8. srpna 2007 Ondřej Šindelář

Obsah

Název práce: Simulátor motokrosu
Autor: Ondřej Šindelář
Katedra (ústav): Katedra aplikované matematiky
Vedoucí bakalářské práce: Mgr. Tomáš Bílý
e-mail vedoucího: tomby@kam.mff.cuni.cz

Abstrakt: Cílem práce je implementace grafické hry simulující jízdu motokrosového jezdce. Uživatel ovládá figurku na motorce tak, aby se dostal ze startu do cíle za co nejkratší čas. Sbírá přitom kontrolní předměty rozmístěné po trati a nesmí narazit hlavou do terénu. Tratě, složené z polygonů, je možné vytvářet a editovat pomocí editoru tratí, který je součástí programu. Kromě hry jednoho hráče je k dispozici hra více hráčů na jednom počítači i po síti ve čtyřech různých režimech. Mimoto lze jednotlivé jízdy ukládat a následně si je přehrávat. Zobrazování scény z bočního pohledu využívá knihovnu OpenGL. Program je implementován pro platformu Win32.

Klíčová slova: motokros, síťová hra, detekce kolize

Title: Motocross simulator
Author: Ondřej Šindelář
Department: Department of Applied Mathematics
Supervisor: Mgr. Tomáš Bílý
Supervisor's e-mail address: tomby@kam.mff.cuni.cz

Abstract: The work aims to implement a graphic game simulating motocross ride. The player controls a motorbike rider and tries to get him from start to finish as quickly as possible. He has to collect checkpoints laid out along a track without striking his head against the ground. Tracks composed of polygons can be created and edited in included track editor. In addition to single-player game there are also four different multiplayer game modes to be played on one or mulitple computers. It is possibile to replay previously saved ride recordings. The scene is displayed as a side view and is rendered using OpenGL library. The program is implemented on Win32 platform.
Keywords: motocross, network game, collisioin detection

MotoX

Simulátor Motokrosu

Část I

Uživatelská dokumentace

Kapitola 1

Možnosti hry

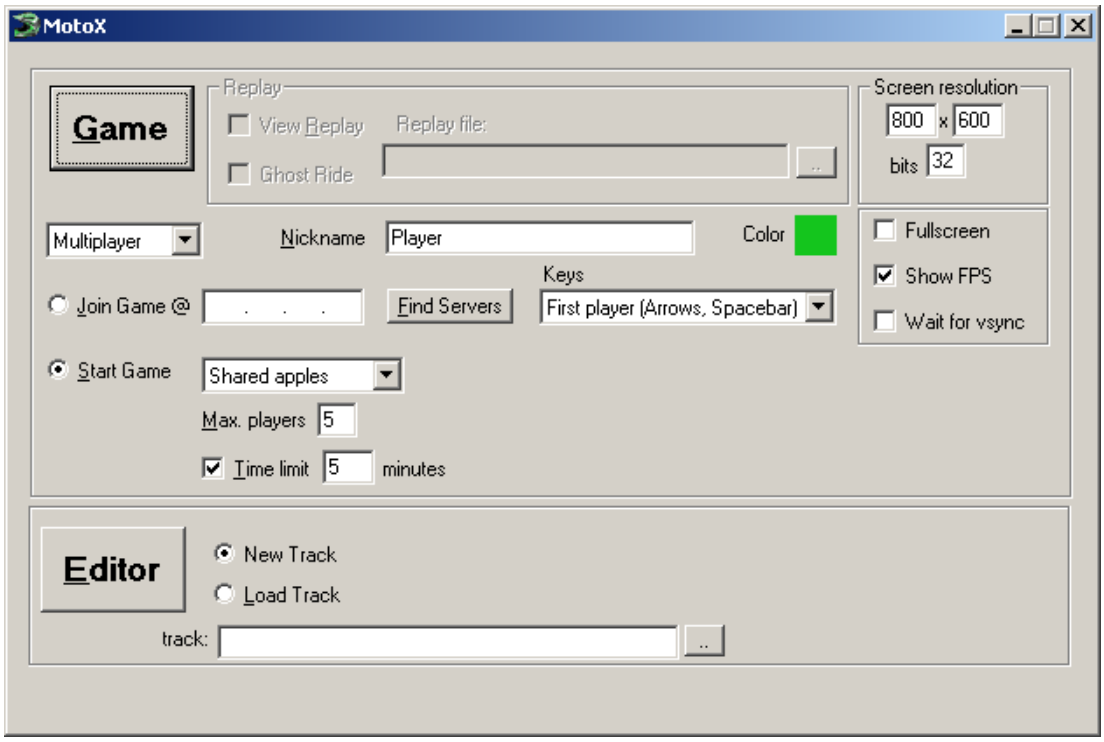
Základním úkolem hráče je pomocí klávesnice ovládat motorku a sbírat jablka rozmístěná po trati, nesmí ale přitom narazit hlavou do země.

Uživatel si může vybrat ze dvou základních typů hry:

- single-player (jeden hráč)
- multiplayer (více hráčů)

K dispozici je také editor tratí, ve kterém je možné tratě vytvářet a upravovat.

Všechny možnosti programu se nastavují pomocí hlavního menu (viz obrázek 1.1).



Obrázek 1.1: Hlavní menu.

Kromě jiného zde můžete nastavit možnosti zobrazování: rozlišení obrazovky (Screen resolution), zobrazení na celou obrazovku (Fullscreen), FPS (počet vykreslených obrázků za sekundu) a čekání na vertikální synchronizaci překreslování (Wait for vsync).

1.1 Single-player

Základní hra

Hra se spustí tlačítkem Game. Hráč si nejprve vybere trať, kterou chce hrát. Po načtení trati se objeví hlavní okno hry a odstartuje se časomíra. Hráč musí posbírat všechna jablka, která trať obsahuje, a poté dojet do cíle. Pokud je to pro vás příliš jednoduché, zkuste to udělat co nejrychleji a překonat nejlepší čas.

Po dojetí do cíle nebo nárazu hlavou do země se objeví menu s možností hru ukončit, restartovat nebo přehrát záznam právě dokončené hry (viz obr. 1.2). Restartovat hru je také možné v průběhu jízdy při pozastavení hry (klávesou Esc).

Během přehrávání záznamu můžete sledovat, které klávesy byly stisknuty, zvyšovat a snižovat rychlost přehrávání a po prohlédnutí je možné tento záznam uložit do souboru. Uložené záznamy se později přehrávají zaškrtnutím View Replay a vyplněním Replay file v hlavním menu.

Obrázek 1.2: Menu ve hře.

Ghost Ride

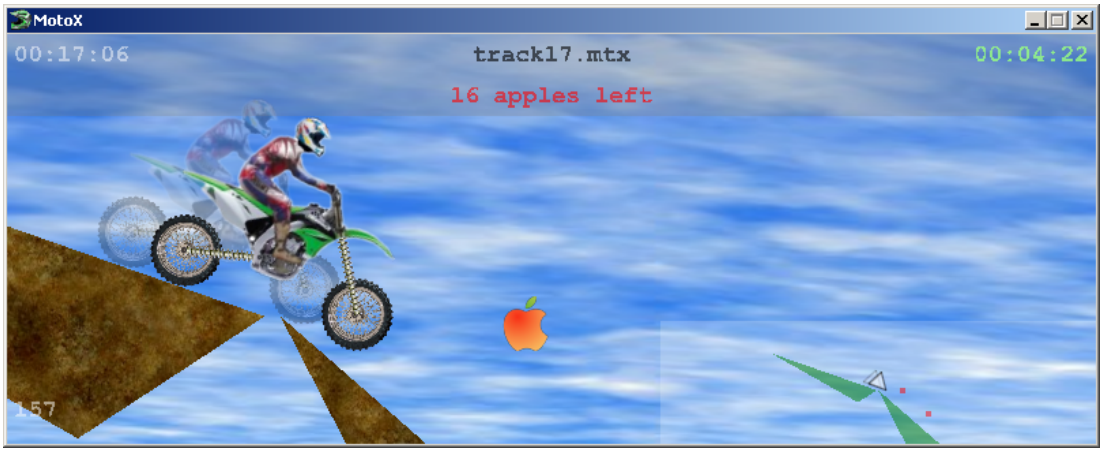
Máte-li nějakou jízdu uloženou, můžete také využít speciální režim hry nazvaný *Ghost Ride* (viz obrázek 1.3). Stačí zaškrtnout Ghost Ride v hlavním menu. V tomto režimu se během hry na pozadí zobrazuje nahraná jízda (duch), takže je možné sledovat pokrok oproti předchozím pokusům.

1.2 Multiplayer

Multiplayer umožňuje hru více hráčů po síti nebo hru dvou hráčů na jednom počítači.

V multiplayeru si můžete vybrat z následujících módů:

- *Normal mode* (základní)
- *Shared apples* (společná jablka)
- *Black Hat* („Na babu“)
- *Black Hat & apples* (kombinace předchozích dvou)



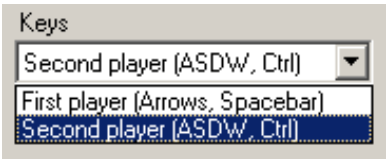
Obrázek 1.3: *Ghost Ride*. Ukazatele stavu hry — nahoře: název trati, počet zbývajících jablek, aktuální a nejlepší čas trati; dole: mapka trati, FPS.

Jeden z hráčů (server) musí založit hru — v hlavním dialogu zaškrtne Start Game, vybere mód hry, stanoví maximální počet hráčů, případně časový limit, a poté vybere trať a čeká na ostatní hráče. Ostatní se připojí k serveru: buď přímo zadají IP adresu jeho počítače nebo pomocí tlačítka Find Servers zobrazí seznam serverů na počítačích ve stejné síti (LAN) a jeden z nich si vyberou. Aby se klienti mohli připojit, musí mít server přístupný port 31072, na kterém naslouchá příchozím spojením.

Každý hráč si zvolí jméno (Nickname) a barvu motorky. Jména musí být unikátní.

Po připojení všech hráčů zakládající hráč (server) hru odstartuje. Je třeba, aby všichni hráči měli na svých počítačích stejnou verzi příslušné tratě.

V případě hry dvou hráčů na jednom počítači se musí program spustit pro každého hráče zvlášť a jeden z nich musí mít v hlavním menu zvoleno, že bude používat klávesy druhého hráče (viz obr. 1.4). Tato položka se automaticky nastaví v programu spuštěném jako druhý v pořadí.



Obrázek 1.4: Výběr kláves.

Normal mode

V tomto módu musí každý hráč posbírat všechna jablka a dorazit do cíle. V případě, že hráč narazí hlavou do země, může začít znovu; čas mu přitom běží dál.

Hra končí, když se všichni hráči dostanou do cíle nebo po uplynutí časového limitu. Vítězí ten, kdo do cíle dorazí nejdřív nebo kdo stihl nasbírat nejvíc jablek do časového limitu.

Obrázek 1.5: Výsledková listina – zobrazí se všem hráčům po skončení hry.

Shared apples

Jablka rozmístěná na trati jsou pro všechny hráče společná. Každé jablko sebere (a dostane za něj bod) ten, kdo k němu přijede jako první. Pokud dosáhne na nějaké jablko v jednom okamžiku víc hráčů najednou, dostane za něj bod každý z nich. V případě, že hráč narazí, může začít znovu, ale jeho skóre se vynuluje. To se stane také po restartu hry pomocí zvolení příslušné položky z menu.

Po vysbírání všech jablek mohou hráči jet do cíle. Za dosažení cíle se příslušnému hráči přičítá bod (jako za každé sebrané jablko). Hra končí po dosažení cíle jedním z hráčů nebo po uplynutí časového limitu. Vyhrává ten, kdo nasbíral nejvíc jablek (viz obrázek 1.5).

Black Hat

Trať je v módu *Black Hat* bez jablek a bez cíle, hráči si pouze předávají černý klobouk, který je na začátku hry náhodně přidělen jednomu z hráčů (jako ve hře Na babu). Hra končí vypršením časového limitu (ten musí být tudíž v tomto módu stanoven) a zvítězí ten, kdo měl na sobě černý klobouk nejkratší dobu. Klobouk se předává dotykem kol nebo hlavy mezi dvěma hráči; kdo předává klobouk, se však nesmí těsně před předáním dotýkat koly ani hlavou jiného hráče. Hráč, který narazí hlavou do země nebo restartuje hru, dostane automaticky klobouk!

Black Hat & apples

Tento mód je kombinací *Black Hat* a *Shared apples*. Hráči se snaží co nejrychleji zbavit černého klobouku jako v módu *Black Hat*, ale rozdíl spočívá v ukončení hry. Kromě vypršení časového limitu může hra skončit také sebráním všech jablek a následným dojetím jednoho z hráčů do cíle. Jablka i cíl jsou totiž na trati obsažena, ale nemají vliv na skóre hráčů – za jejich získání se nepřidělují body. Mohou ale pomoci k vítězství hráče, který má náskok před ostatními.

Kapitola 2

Ovládání hry

2.1 Single-player

Pro zobrazení základních ovládacích kláves během hry stiskněte klávesu F1 nebo vyberte položku Help z menu ve hře.

Při hře:	
šipka NAHORU	plyn
šipka DOLŮ	brzda
šipka VLEVO	otáčení motorky doleva (proti směru hodinových ručiček)
šipka VPRAVO	otáčení motorky doprava (po směru hodinových ručiček)
mezerník	změna orientace motorky (otočí motorku tak, aby mohla jet na druhou stranu)
Esc	pozastavení hry
M	zobrazení/skrytí mapy
Enter	výběr položky menu

Při přehrávání záznamu:	
šipka NAHORU	zvýšení rychlostí přehrávání
šipka DOLŮ	snížení rychlostí přehrávání
mezerník	pozastavení přehrávání

2.2 Multiplayer

Více hráčů (po síti):	
N	zobrazení/skrytí jmen na motorkách hráčů
Tab	zobrazení průběžných výsledků

K ovládání motorky a menu se používají stejné klávesy jako v single-playeru. Po dojetí do cíle se můžete šipkami pohybovat po trati a sledovat ostatní hráče. Klávesou Tab přepínáte mezi hráči, kteří ještě nedojeli, a můžete sledovat jízdu z jejich pohledu.

Dva hráči na jednom počítači:	
Pro zobrazení jmen a průběžného pořadí fungují stejné klávesy (pro oba hráče společné) jako při hře po síti. K ovládání motorky a menu má každý hráč své vlastní klávesy.	
První hráč:	
První hráč používá stejné klávesy jako při hře jednoho hráče.	
Druhý hráč:	
W	plyn
S	brzda
A	otáčení motorky doleva (proti směru hodinových ručiček)
D	otáčení motorky doprava (po směru hodinových ručiček)
Ctrl	změna orientace motorky (otočí motorku tak, aby se dalo jet na druhou stranu)
Shift	výběr položky menu

2.3 Editor

Editor (obrázek 2.1) má několik módů. Na začátku je to polygon-mód (editace polygonů): levým tlačítkem myši přidáváte vrcholy k aktuálnímu polygonu; klávesami Page Up a Page Down vybíráte aktivní vrchol polygonu, který se dá smazat klávesou Del; aktivní vrchol také určuje místo vkládání dalších vrcholů; mezerníkem můžete obrátit směr přidávání vrcholů; pravým tlačítkem myši polygon ukončíte. Stejným tlačítkem (nebo klávesou M) můžete polygon přesunout. Kopii aktivního polygonu můžete vytvořit klávesou F5. Kombinace kláves Ctrl+M, resp. Ctrl+F převrátí aktivní polygon horizontálně, resp. vertikálně. Klávesa E umožňuje znovu upravit již ukončený polygon.

Když stisknete klávesu Tab, aktivujete apple-mód: levým tlačítkem myši můžete přidávat jablka a pravým je posouvat.

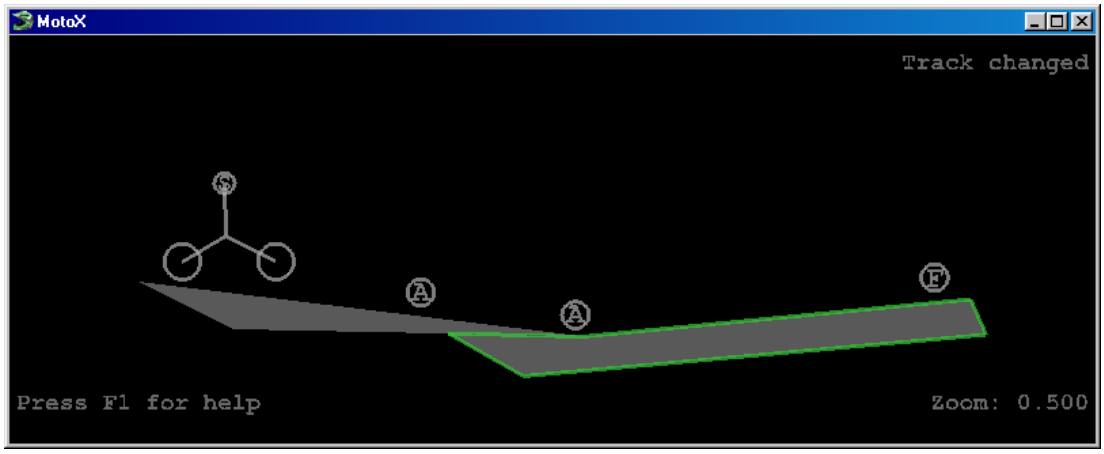
Pokud stisknete klávesu S, můžete přesunout místo startu; podobně klávesou F můžete přesunout místo cíle. Při polygon-módu nebo apple-módu se dá vybrat aktivní objekt klávesami PageDown a PageUp a smazat klávesou Delete.

Šipkami se můžete pohybovat po trati a klávesami + a - nebo kolečkem myši trať přibližovat a oddalovat (se Shiftem po menších skocích). Když stisknete levé tlačítko myši spolu s klávesou Ctrl, přesune se střed obrazovky na místo kurzoru. Kombinace kláves Ctrl+S je určena pro uložení trati. Pro opuštění editoru stiskněte klávesu Esc. Pokud si tyto klávesy nepamatujete, můžete si je zobrazit klávesou F1.

Přepsáním existující tratě smažete také její nejlepší čas!

Klávesy pro ovládání editoru jsou shrnuté v následující tabulce.

Ovládání editoru:	
F1	zobrazí nápovědu
šipky	přesouvání pohledu
+/- (kolečko myši)	přibližování/oddalování
Ctrl + LTM[1]	přesune střed obrazovky na místo kurzoru
Tab	přepínání mezi polygon- a apple-módem
PgDown	další polygon/jablko/vrchol
PgUp	předchozí polygon/jablko/vrchol
Del	smaže aktivní polygon/jablko/vrchol
M nebo PTM[2]	přesunutí aktivního polygonu/jablka
F5	zkopíruje aktivní polygon
E	editace aktivního polygonu
Mezerník	otočení směru editace polygonu
Ctrl+F	převrácení aktivního polygonu vertikálně
Ctrl+M	převrácení aktivního polygonu horizontálně
S	přesunutí startu
F	přesunutí cíle
LTM	přidá vrchol/přidá jablko/umístí start nebo cíl (záleží na aktuálním módu)
PTM	uzavře polygon (při vytváření polygonu)
Ctrl+S	uložení trati
Esc	ukončení editoru (uložení trati)



Obrázek 2.1: Editor tratí. Zeleně orámovaný je aktivní polygon.

[1]LTM – levé tlačítko myši
[2]PTM – pravé tlačítko myši

Část II

Programátorská dokumentace

Kapitola 3

Stručný popis

MotoX je hra, která se snaží simulovat jízdu motokrosového jezdce. Uživatel pomocí klávesnice naviguje figurku na motorce různými tratěmi sestavenými z polygonů, které si může vytvořit v zabudovaném editoru. Kromě hry jednoho hráče je k dispozici hra více hráčů na jednom počítači i po síti ve čtyřech různých módech a přehrávání uložených záznamů jízd. Grafika se zobrazuje pomocí OpenGL (viz [4]) kvůli akceleraci vykreslování a jednoduché práci s geometrickými primitivy; k načítání obrázků (textur) je použita knihovna BMGLib (viz [3]). Při překladu je proto potřeba přilinkovat knihovny opengl32.lib, glu32.lib a BMGLib.lib; dále WS2_32.lib a comctl32.lib kvůli podpoře síťové komunikace a rozšířených ovládacích dialogových prvků. Program je implementován v jazyce C/C++ pro běh na platformě MS Windows.

Program obsahuje dva hlavní celky:

- editor tratí — uživatel v něm sestavuje z polygonů a jablek trať; nakonec ji může uložit do souboru
- samotnou hru — pokud možno reálně simuluje pohyb motorky s jezdcem v tomto vytvořeném světě

Program se svými vlastnostmi snaží nabízet uživateli zábavu a komfort při jeho užívání. Má jednoduché ovládání, které však dovoluje velkou variabilitu pohybu ve světě hry; zjednodušeně také zobrazuje realitu (svět je dvourozměrný), ale zároveň se snaží co možná nejrealističtěji simulovat reakce mezi motorkou a terénem. Nabízí uživateli úkoly vyzývající k pokoření: dojet do cíle, překonat nejlepší čas trati nebo srovnat své schopnosti ovládání motorky s jinými hráči ve hře více hráčů současně (multiplayer). Hráč také může své výkony zlepšovat s pomocí zaznamenávání svých pokusů a jejich následných přehrávání nebo v reálném čase sledovat pokrok ve speciálním režimu hry nazvaném *Ghost Ride*, ve kterém se záznam přehrává na pozadí normální jízdy.

Většina entit ve světě hry je reprezentována objekty hierarchicky sestavených tříd. Každá trať (třída Ctrack) obsahuje startovní a koncovou pozici (třída Cvector), seznam polygonů (třída Cpolygons), seznam checkpointů — jablek (třída Capples). Každý polygon (třída Cpolygon) je realizován seznamem svých vrcholů, ty jsou stejně jako všechny 2D vektory, rychlosti a pozice reprezentovány objekty

třídy Cvector. Objekt hráče (třída Cmoto) má svojí pozici, rychlost, úhel otočení a úhlovou rychlost, dále dvě kola a hlavu jako objekty třídy Cball, která obecně reprezentuje pohybující se a otáčející se disk s nějakým poloměrem. V průběhu každé hry se ukládá záznam stisknutých kláves pomocí třídy Crecorder umožňující pozdější přehrávání záznamu.

Pro multiplayer je navržena třída CmultiPlayer, které je reprezentací hráče. Obsahuje informace o hráči, například jako jeho jméno, objekt jeho motorky, stav připojení, skóre a další. Server využívá instance rozšířeného potomka této třídy (CmultiPlayerClient) pro uložení potřebných informací o klientech (socket spojení, handle obslužného vlákna).

Kapitola 4

Algoritmus simulace

4.1 Idea

Základem algoritmu je předpoklad, že výsledek simulace bude při daném uživatelském vstupu po daném čase stejný nezávisle na podmínkách běhu programu (zátěž, výkon počítače). Tento předpoklad je nutné požadovat jednak proto, že je zcela přirozený, a také kvůli fungování částí programu, jako je záznam jízdy nebo multiplayer. Jediné možné řešení je přepočítávat simulaci v krocích v předem dané konstantní frekvenci. Experiment potvrdil, že 100 Hz je rozumná hodnota, protože nezatěžuje nadměrně procesor a zároveň je simulace dostatečně plynulá. Měření času mezi přepočítáváními zajišťuje vysokofrekvenční čítač, který je součástí služeb operačního systému.

Těsně před zobrazením výsledku simulace do okna se vždy spočítá ještě jeden její mezikrok podle přesné doby od posledního pravidelného přepočítávacího kroku. Důvodem k tomu je požadavek zajištění dokonalé plynulosti animace pohybů.

Přepočítávání i zobrazování se vykonávají ve speciálních vláknech pro oddělení funkčnosti a synchronní nezávislosti obou procesů a také za účelem využití výkonu moderních vícejádrových procesorů. Prvotní vlákno procesu se stará o zpracování zpráv okna. Kvůli exkluzivnímu přístupu ke sdíleným datům se musí zajistit mezivláknová synchronizace.

4.2 Přepočítávání

Při přepočítávání se uplatní fyzikální zákony, hlavně

$$v = \frac{ds}{dt}, \tag{4.1}$$

tedy vektor rychlosti je roven derivaci dráhy podle času. My budeme počítat v dostatečně malých časových úsecích, takže můžeme psát

$$s = v \cdot t, \tag{4.2}$$

čili pokud známe vektor rychlosti nějakého předmětu, můžeme spočítat vektor posunutí vynásobením této rychlosti časem, který uběhl od minulého přepočtu. Uvažujeme motorku jako tři hmotné předměty — dvě kola a samotnou motorku s jezdcem, z nichž každý má uloženou svoji polohu v trati a rychlost pohybu jako dvousložkové vektory, dále úhel otočení a rotační rychlost jako skaláry. Takže podle vzorce (4.2) upravíme polohu všech tří částí motorky podle jejich aktuálních rychlostí, obdobně změníme úhly otočení podle rotačních rychlostí.

4.3 Detekce kolizí

Dále musíme zjistit, zda nenastala posunem nějaká kolize kol motorky nebo hlavy jezdce s terénem tratě, s jablky (kontrolní předměty) nebo s cílem, a v takovém případě na takovou situaci adekvátně zareagovat. K tomu slouží algoritmus detekce kolize popsaný např. v [1], který používá zákonitosti plošné geometrie. Při kolizi s jablky se příslušné jablko odstraní a sníží se počet zbývajících jablek. Hra se ukončí při kolizi hlavy s terénem (neúspěšně) nebo kolizí hlavy nebo kol s cílem, pokud již nejsou žádná jablka (úspěšně).

Testování kolize s jablkem nebo cílem je jednoduchý test nerovnosti vzdálenosti středu kola (hlavy) a jablka (cíle) a součtu jejich poloměrů.

Kolize kol s terénem a reakce na ni je zdaleka nejtěžší část výpočtu simulace. Základem je detekce kolize kola s hranou polygonu: pokud kolo nekoliduje ani s jedním vrcholem hrany (prostý test vzdálenosti od středu kola), znamená to, že ho hrana protíná na dvou místech nebo vůbec. V tomto případě kolize nastala, právě když je kolmý průmět středu kola v mezích hrany a jeho vzdálenost středu od hrany je menší než poloměr kola. To lze spočítat použitím jednoduché geometrie vektorů.

Pokud nastala nějaká kolize, najde se nejdřív bod první kolize: postupně se promítne trajektorie pohybu kola od posledního přepočítávání přes všechny hrany všech polygonů, najde se jejich nejvzdálenější průsečík a kolo se posune na místo kolize. Potom se změní jeho rychlost podle dosavadní rychlosti, rotace kola a úhlu dopadu na kolizní hranu (při kolizi s vrcholem polygonu je hrana kolize tečna kola v místě kolize); dále se změní rotační rychlost podle průmětu odrazové posuvné rychlosti na hranu kolize. Kolo se potom posune podle odrazové rychlosti a čas, o který jsme se museli vrátit na místo kolize, a opakujeme detekci kolize a reakci na ni, dokud se nedostaneme do bezkolizní situace. Ta určitě nastane po malém počtu takových cyklů, pokud je odraz dostatečně ztrátový — velká část dopadové energie se ztratí. Schéma nalezení bodu kolize je na obrázcích 6.3 a 6.4.

4.4 Aplikace sil

Jakmile už motorka nebude kolidovat s terénem, bude možné aplikovat síly, kterými působí tělo motorky na kola vlivem pružin. Nejdřív zjistíme velikost těchto sil podle upraveného Hookova zákona

$$F = -k \cdot x \cdot \log\left(|\,x\,| + 1\right), \tag{4.3}$$

kde x je velikost vychýlení pružiny. Stejnou silou působí také obě kola na celou motorku podle Newtonova zákona akce a reakce. Síly působící mezi hmotnými tělesy se přemění na zrychlení podle druhého Newtonova zákona

$$F = m \cdot a. \tag{4.4}$$

Takto získané vektory zrychlení se vynásobí časem uplynulým od posledního přepočtu a přičtou se k vektorům rychlosti podle definice zrychlení

$$a = \frac{dv}{dt}. \tag{4.5}$$

4.5 Shrnutí přepočítávání

- vyhodnocení klávesového vstupu od uživatele
- posunutí každé části motorky (kol a samotné motorky) podle jejich rychlosti a času, který uběhl od minulého přepočítávání
- odraz kol od terénu pomocí detekce kolize disku s úsečkou a reakce na ni (pomocí vektorové geometrie)
- spočítání sil mezi koly a motorkou způsobených pružinami
- aplikace těchto sil na zrychlení kol a motorky, zrychlení se projeví ve změně rychlosti
- zjištění dosažení checkpointů (jablek) a cíle podle detekce kolize disk–disk
- zjištění nárazu (kolize hlavy s terénem), detekce kolize kol s terénem

Kapitola 5

Multiplayer

Hra pro více hráčů má za cíl propojit více uživatelů do jedné společné hry a to tak, aby se poloha a stav každého hráče přenášely a zobrazovaly všem ostatním hráčům pokud možno v reálném čase. Zároveň nastavuje jiná pravidla závislá na zvoleném módu hry a hráčům nabízí možnost soutěžit s ostatními.

5.1 Architektura

Hra více hráčů je založená na architektuře klient–server, což přináší několik výhod. Jelikož mají hráči na výběr mnoho možností hry, musí existovat autorita (server), která pro jednu konkrétní hru všechny tyto možnosti nastaví. Je to hlavně trať, na které se bude hra konat, potom také mód hry, maximální počet hráčů ve hře, případně časový limit. Server také zajišťuje korektní distribuci a jednotnost informací sdílených mezi hráči a konečně má na starosti přepočítávání informací týkajících se jednotlivých módů hry více hráčů (např. přidělování skóre hráčům, ukončení hry).

Jakmile server založí hru, začne vysílat pakety (broadcast), které informují klienty o tom, kam se mohou připojit. Po připojení hráče – klienta si oba vymění potřebné informace o hře, jako je název trati, mód hry, klientovo jméno a seznam ostatních hráčů. Pro zajištění alespoň základní bezpečnosti a důvěryhodnosti komunikace se ověřuje shodnost kontrolních součtů trati a samotného programu. Když to hráč – server uzná za vhodné, zahájí hru.

Komunikace v samotné hře závisí na zvoleném módu (viz sekci 5.5), ale v každém případě ji vždy zahajuje klient zasláním svých údajů (hlavně aktuální polohy, rychlosti pohybu a stisknutých kláves) a následuje odpověď serveru s údaji svými a ostatních hráčů. Toto vyměňování informací by mělo probíhat co nejčastěji, ale maximálně s frekvencí přepočítávání, vyšší by neměla smysl, protože by se posílala vícekrát stejná data. V praxi je to vyřešeno manuálními objekty události (manual-reset event object). Klientovo přepočítávací vlákno po skončení každého přepočítávání nastaví stav event objektu na signalizovaný; na to čeká síťové vlákno, které okamžitě pošle informace o hráči serveru; následně síťové vlákno serveru pošle zpět informace o ostatních hráčích a případně další potřebná data.

5.2 Protokol TCP

Program ke komunikaci využívá síťové protokoly TCP/IP implementované v operačním systému Microsoft Windows — knihovnu Winsock. Konkrétně byl použit protokol TCP, protože zajišťuje spolehlivý přenos dat, který je nutný pro správné fungování hry více hráčů. Pro zajištění menší latence spojení v méně spolehlivých sítích by mohl být použit protokol UDP. Implementace by však byla mnohem náročnější, protože spolehlivost komunikace je nutná alespoň pro přenášení změn stavů hry (např. začátek, konec hry, změna skóre hráčů), který by jinak nemusel být konzistentní mezi hráči. Program je také koncipován spíše pro rychlé lokální sítě s nízkým přenosovým zpožděním, které nebudou tolik zatěžovány síťovým provozem hry a kde tedy nebude vadit režie protokolu TCP nutná k zajištění spolehlivosti.

Způsobem, jak zrychlit odezvu TCP spojení, je zakázat používání tzv. Nagleova algoritmu, který se snaží slučováním malých zpráv a čekáním na potvrzení posledního odeslaného segmentu snížit zatížení sítě. Tento algoritmus by výrazně prodlužoval zpoždění komunikace mezi serverem a klienty, jehož velikost je při hře kritická pro plynulost animace.

Síťová část multiplayeru je implementována v souboru Networking.cpp — viz sekci 6.6.

5.3 Lokální multiplayer

Pro hru více uživatelů na jednom počítači bylo využito stávající funkcionality, a to tak, že se k sobě lokální hráči chovají, jako kdyby komunikovali po síti mezi oddělenými počítači. Kvůli prostorovým možnostem byl omezen počet lokálních hráčů na dva. Každý hráč tedy musí mít spuštěnu vlastní instanci programu, která bude zobrazovat hru z jeho pohledu. Je tu ale problém uživatelského vstupu: oba hráči nemohou ovládat program stejnými klávesami najednou a navíc při užití standardního klávesového vstupu pomocí zpráv operačního systému bude o vstupu vědět vždy pouze jedno z oken. Řešením je definovat pro jednoho z hráčů jiné klávesy a používat systémovou funkci pro asynchronní zjišťování stavu kláves (GetAsyncKeyState). I tak jsou ale některé funkce, jako například celoobrazovkový režim, samozřejmě omezeny.

5.4 Časová synchronizace

Zřejmým problémem při síťové komunikaci je rozcházení hodin hráčů. I při využití vysokofrekvenčních čítačů se může po čase objevit rozdíl v čase vedoucí k zobrazování neaktuálního nebo chybného stavu hry. Tento problém je v programu řešen průběžným opravováním hodin klientů podle času serveru. Klienti posílají kromě jiných informací údaj o čase (dobu od začátku hry) těsně před odesláním paketu, server spočítá rozdíl vztažený ke svým hodinám a pošle ho spolu se svým časem a ostatními informacemi zpátky klientovi. Klient si stejně

spočítá rozdíl času serveru oproti svým hodinám a pevně danou část průměru těchto rozdílů použije jako korekci svých hodin. Důvodem pro počítání rozdílů na obou stranách je možné přenosové zpoždění, jehož vliv (pokud je dostatečně konstantní) se zprůměrováním vyruší. Čas se opravuje pouze částí rozdílu, aby při krátkodobých výpadcích nebo kolísání latence nepoškodil jinak správný časový údaj. Při korekcích u každého přepočítávání se i tak dobře opravují i velké nerovnosti frekvencí čítačů.

5.5 Módy multiplayeru

Normal mode

V tomto módu se každý hráč snaží stejně jako v single-playeru sebrat všechna jablka a poté co nejdříve dojet do cíle. Hru vyhraje, pokud to stihne v kratším čase než ostatní hráči. Každý klient si sám přepočítává svůj stav a skóre (počet sebraných jablek) a posílá tyto údaje serveru. Ten je pouze přeposílá ostatním.

Shared apples

Jablka jsou zde sdílená, takže za ně dostávají body pouze ti hráči, kteří jich dosáhnou první. Hra skončí pro všechny hráče ve stejný okamžik, a to při dosažení cíle po vysbírání všech jablek. To znamená pro server práci navíc, jelikož musí po každém vlastním pravidelném přepočítávání kontrolovat, jestli nějaký hráč nedosáhl na některá z jablek nebo nakonec na cíl. Kdyby tyto výpočty prováděl každý sám za sebe, nemusel by být výsledek u všech hráčů stejný. Pověřením serveru tímto úkolem a následnou distribucí ostatním hráčům se tento problém vyřeší. Server ovšem nemusí mít úplně aktuální informace o poloze všech klientů kvůli zpoždění síťovým přenosem, takže si je dopočítá podle posledních známých informací o rychlostech a stisknutých klávesách. To může vést k chybě oproti skutečnému stavu hráčů, ale je to nejlepší aproximace, jakou můžeme udělat, aniž bychom ztratili možnost simulace v reálném čase, a přitom je tato chyba zanedbatelná.

Server posílá klientům navíc postupně informace o odebraných jablkách (v každé odpovědi klientovi maximálně jedno jablko) a každému klientovi ještě jeho vlastní skóre (v *Normal mode* si skóre přepočítaval každý klient sám). Naopak klienti místo svého skóre posílají pouze speciální hodnotu (-1) v případě, že narazili a má se jim vynulovat skóre.

Black Hat

Hráči v tomto módu nesbírají jablka, ale předávají si černý klobouk. Předávání kontroluje server, stejně jako v módu *Shared apples*. Server zjistí kolize mezi koly nebo hlavou držitele klobouku a koly nebo hlavou ostatních hráčů za podmínky, že při minulé kontrole žádná taková kolize nenastala. Při každé takové kolizi předá klobouk kolidujícímu hráči.

Server posílá klientům jako skóre celkový čas držení klobouku každého hráče (i jich samotných). Podle zvýšení skóre oproti minulého stavu tak klienti poznají, kdo klobouk dostal. Stejně jako v předchozím módu, klienti posílají serveru speciální hodnotu místo svého skóre, pokud narazili a má se jim předat černý klobouk.

Black Hat & apples

Tento mód je kombinací předchozích dvou módů: hráči si předávají klobouk (stejně jako v *Black Hat*) a sbírají jablka (jako v *Shared apples*). Za jablka ale nedostávají body, skóre a konečné pořadí se počítají podle celkového času držení klobouku.

Hra může skončit dvěma způsoby: buď se vysbírají všechna jablka a někdo dojede do cíle, nebo po vypršení časového limitu jako v ostatních módech.

Data posílaná klientům zahrnují jejich skóre (doba, jakou měli na sobě klobouk) a informace o sebraných jablkách. Klienti posílají stejná data jako v módu *Shared apples*.

Kapitola 6

Obsah jednotlivých modulů a jejich popis

6.1 globals.cpp

- obsahuje globální proměnné a konstanty pro celý program
- nejdůležitější z nich:
  - Dt — časový úsek mezi přepočítáváními hry (0,01 s)
  - isGame — určuje, v které části programu se nacházíme (true – hra, false – editor)
  - keys — pole stisknutých kláves, mění se podle poslaných klávesových zpráv
  - multiplayer — určuje, zda je aktivní hra více hráčů
  - recounts — počet přepočtů od začátku hry
  - currTime — doba jízdy v centisekundách (platí, že recounts = currTime)

6.2 main.cpp

- obsahuje vstupní bod programu (funkce WinMain), zobrazuje menu — dialog s ovládacími prvky pro nastavení hry, spouštění editoru a hry
- k provozu dialogu je použito standardní Windows API — vytvoření funkcí CreateDialogParam pomocí šablony z dialogového zdroje, procedura okna je MenuDialogProc
- funkce SaveConfig, ReadConfig — načítání a ukládání konfigurace (údaje nastavené v dialogu) do souboru MotoX.cfg; konfigurace se automaticky načte při vytvoření dialogu a uloží při jeho ukončení
- funkce GetDlgValues a SetDlgValues pro nastavení a získání údajů dialogu

- funkce OpenFileDialog a SaveFileDialog pro zobrazování dialogů na vybírání souborů pro uložení nebo čtení
- funkce EnableSingleMultiPlayer — při změně hry ze single-player na multiplayer nebo opačně zruší možnost ovládat prvky patřící k opačnému módu; obdobně je to se změnou Start/Join game a obslužnou funkcí EnableStartJoinMultiPlayer
- stisknutí tlačítka Find Servers spustí další dialog zobrazující nalezené servery se založenou hrou; vytvoří se funkcí DialogBox a zprávy pro něj zpracovává procedura ServersDialogProc; ta také pravidelně kontroluje příchozí broadcastové pakety s informacemi o založených hrách (funkcí FindServer) a vyplňuje je jako položky do seznamu dialogu
- funkce CountFileChecksum spočítá kontrolní součet programu (pro ověření stejné verze mezi všemi hráči v multiplayeru); jsou tím vyloučeny chyby v komunikaci v důsledku rozdílného aplikačního protokolu různých verzí programu

6.3 GameWindow.cpp

GameOpenGLWinMain()

- hlavní procedura samotné hry
- pomocí rutin Windows API vytvoří okno, vlákna pro vykreslování a přepočítávání a pak vstoupí do hlavního cyklu hry
- v hlavním cyklu se vyzvedávají a směrují ke zpracování zprávy systému (např. stisk kláves)
- po skončení hry se počká na ukončení ostatních vláken
- nakonec se zruší vykreslovací okno (KillGLWindow) a uvolní se všechny proměnné použité ve hře (GameEnd)

RecountThread()

- procedura přepočítávacího vlákna
- zde se přepočítává stav hry (GameRecount) v přesně stanovené frekvenci — zajištěno vysokofrekvenčním čítačem, jehož stav zjišťuje dvojice systémových funkcí QueryPerformanceCounter a QueryPerformanceFrequency
- přepočet se vykonává, dokud je čas stanovený pro další přepočet (určený proměnnou NextRecountTime) nižší než aktuální čas daný čítačem
- po každém přepočtu se stanoví čas dalšího přičtením konstantní periody přepočítávání Dt nastavené na 0,01 s

- kvůli konzistenci přepočítávaných dat k nim potřebujeme zajistit exkluzivní přístup při jejich zápisu — to je vyřešeno použitím semaforu motoPlayerLockSemaphore:
  - na začátku je jeho čítač nastaven na počet vláken, která budou data pouze číst — to jest jedno renderovací vlákno a v multiplayeru navíc počet síťových vláken (u klienta jedno, u serveru odpovídá počtu klientů)
  - každé vlákno, které bude chtít kritická data číst, sníží čítač o jedničku a po skončení čtení ho zase o jedničku zvětší — tím je umožněno simultánní čtení
  - před zápisem dat (v přepočítávacím vlákně RecountThread) se sníží čítač až na nulu — pokud některá vlákna data zrovna čtou, toto vlákno se zablokuje do té doby, než bude kritická sekce volná
  - během přepočítávání je znemožněno data číst, protože semafor je zablokován; po skončení přepočítávání je čítač vrácen na počáteční hodnotu, čímž se uvolní kritická sekce pro ostatní vlákna
- výsledek přepočítávání je jedna z následujících hodnot:
  - result_crashed (hráč narazil hlavou do země)
  - result_finished (hráč skončil – dojel do cíle)
  - result_normal (normální stav),

  což jsou hodnoty výčtového typu GameRefreshResult (další hodnoty jsou použity v obnovovací proceduře GameRefreshScreen — viz sekci 6.4); tento stav hry se uloží do globální proměnné RecountResult pro využití kreslícím vláknem
- při hře více hráčů se zde také kontroluje časový limit
- server provádí další činnosti související s aktuálním módem: kontrola sdílených jablek v *Shared apples* (playersCheckApplesFinish), kontrola předání klobouku v módu *Black Hat* (checkBlackHat) nebo obojí v módu *Black Hat & apples* (viz sekci 6.6)
- klient sděluje svému síťovému vláknu, že má poslat aktuální stav přepočítávání serveru — pomocí nastavení event objektu (SetEvent)

RenderThread()

- procedura vykreslovacího vlákna
- zde se nejprve inicializuje hra: získá se device context vytvořeného okna a vytvoří se OpenGL rendering context; nainicializují se všechny proměnné potřebné pro hru (načtení tratě, záznamu, textur — pomocí GameInit)

- při multiplayeru se také inicializuje síťová komunikace vytvořením síťových vláken: u serveru je to vlákno přijímající klientská spojení (procedura NetworkServerAcceptThread) a vlákno vysílající informace o serveru (BroadcastThread); u klienta celou komunikaci obstarává jedno vlákno (NetworkClientThread)

- inicializace sítě se musí nastavit až po načtení hry (trať musí být korektně načtená) a její součástí je načítání textur, které musí být vykonáno ve stejném vlákně, ve kterém se potom renderuje — proto se inicializace hry provádí zde (více informací o síťové komunikaci v sekci 6.6)

- po inicializaci se zobrazí okno a nastartuje přepočítávání (vzbudí se přepočítávací thread)

- dále se vstoupí do hlavního cyklu procedury, ve kterém se obnovuje grafický výstup hry funkcí GameRefreshScreen

- funkce GameRefreshScreen vrací aktuální stav hry (hodnota výčtového typu GameRefreshResult), jehož změna může vyžadovat adekvátní reakce: po zvolení restartu hry nebo přehrání záznamu se musí znovu načíst všechny proměnné hry, po zvolení položky Exit musíme hru ukončit

- po ukončení hry se uvolní rendering a device context (KillRenderingContext), pokud je aktivní multiplayer, ukončí se síťová komunikace funkcí NetworkFinish

StartTimer()

- stanoví čas dalšího přepočtu na určitý čas od daného okamžiku — například při startu hry se nastaví časová mezera 0,5 s, aby nebyl start náhlý

6.4 GameRefreshScreen.cpp

GameRefreshScreen()

- obnovuje grafický výstup hry v okně, zajišťuje zobrazování menu ve hře a jeho ovládání

- nejdřív se spočítá aktuální pozice hráče podle času od posledního pravidelného přepočítávání pomocí funkce RecountBeforeRender

- funkce GameRender zajistí překreslení pozadí, terénu, jablek a cíle, motorky s jezdcem a mapky (viz sekci 6.5)

- dále se vypíše jméno tratě, čas od začátku hry (currTime), popřípadě nejlepší čas (bestTime), čas záznamu, počet zbývajících jablek (appleC), spočítá se a napíše FPS

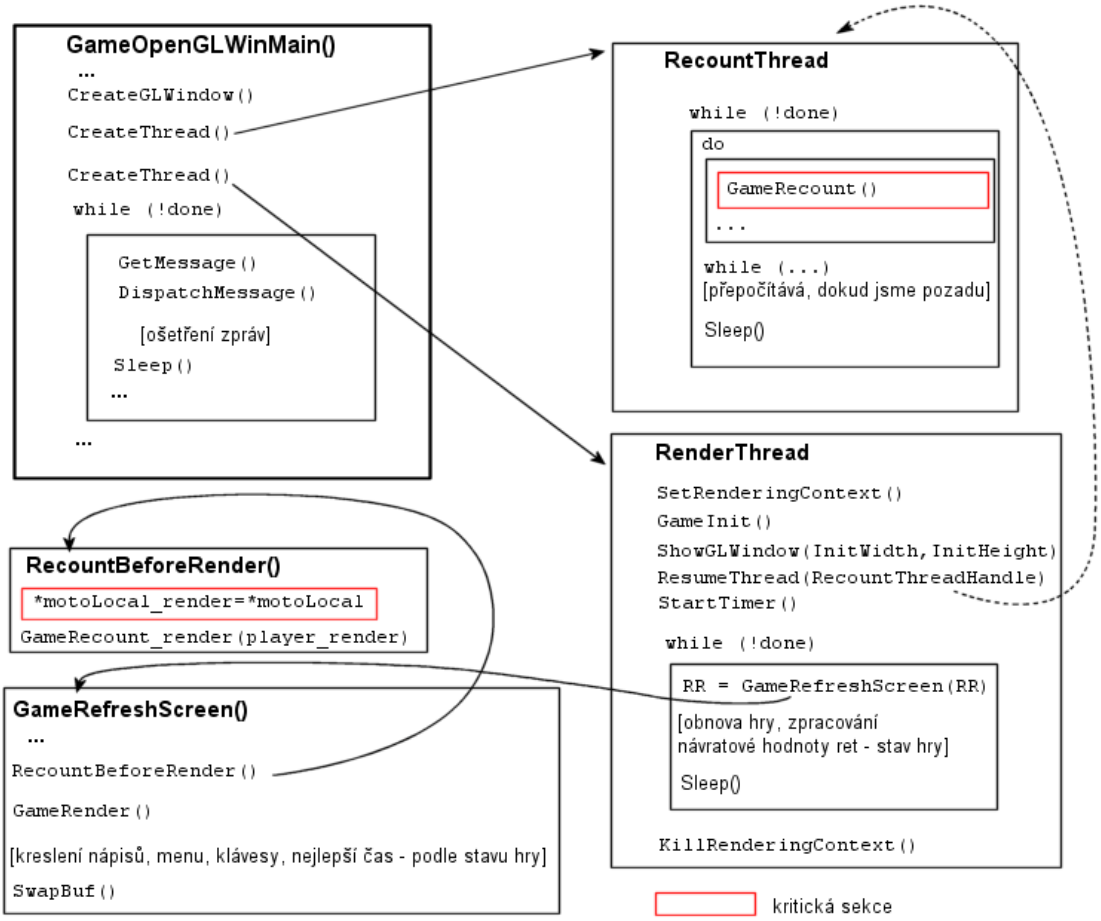
- obstarává in-game menu, které je reprezentováno třídou gameMenuClass: při pozastavené nebo skončené hře (po nárazu, dojetí do cíle) a před začátkem hry více hráčů zobrazí stav hry a seznam dalších možných reakcí uživatele, zpracovává stisknuté klávesy
- návratová hodnota je typu GameRefreshResult a určuje stav hry; je dána jednak jejím předchozím stavem (předaným funkci jako parametr), jednak posledním výsledkem přepočítávacího vlákna (RecountResult) a také ji může změnit uživatel výběrem položek z menu (result_restart, result_replay, result_exit)
- při multiplayeru se navíc zobrazují informace o pořadí lokálního hráče a zbývajícím počtu hráčů, kteří ještě nedojeli do cíle; po skončení nebo při stisku klávesy Tab se zobrazuje tabulka pořadí všech hráčů pomocí funkce list_players_ingame; pořadí hráčů se ale musí vždy nejprve seřadit funkcí sort_ingame_list
- před zahájením hry více hráčů se zobrazuje speciální menu se seznamem připojených hráčů (funkce list_players_pregame)

class gameMenuClass

- reprezentuje menu zobrazované na obrazovce během hry
- obsahuje pole položek, které se vyplní při vytváření menu metodou create; dále počet položek, aktivní položku, typ menu: Paused, Finished, Game Over, Multiplayer startup
- metoda draw nakreslí na obrazovku menu podle daného typu s označenou aktivní položkou
- metoda process zpracuje požadavek na výběr položky z menu a vrátí změněný stav hry (typu GameRefreshResult)

RecountBeforeRender()

- spočítá aktuální polohu všech motorek na trati (lokálního hráče, ducha v režimu *Ghost Ride*, ostatních hráčů v multiplayeru)
- používá přepočítávací funkci uzpůsobenou pro překreslování (nepřepočítává kolize s jablky a cílem, neobrací motorku) — GameRecount_render; posouvá motorky o čas, který uběhl od posledního pravidelného přepočítávání
- hráči v multiplayeru mohou být pozadu o více než jen jeden krok, takže kvůli korektnímu přepočítávání se nejdřív dopočítají jejich zpožděné pravidelné přepočty



Obrázek 6.1: Diagram běhu programu — hlavní smyčka, přepočítávání a rendering.

6.5 Game.cpp

GameRecount()

- přepočítávání stavu motorky; vrací hodnotu z výčtu GameRefreshResult (normální stav je result_normal)
- vyhodnotí stisk kláves spuštěním procedury EvalKeys
- posune a otočí kola motorky o Dt-násobky jednotlivých rychlostí (RecountWheel) (podle vzorce 4.2) a zkontrolují se kolize kol s polygony (WheelCollisions)
- obdobně posune a otočí tělo motorky s jezdcem
- pomocí metody collision objektu motorky zjistí, jestli nenastala kolize hlavy s terénem (v tom případě vrátí result_crashed)
- dále zjistí kolize hlavy nebo kol jezdce s polygony a jablky – checkpointy,

případně s cílem, pokud žádná jablka nezbývají (při dokončení hry funkce vrací result_finished)

- vyhodnotí síly mezi koly a zbytkem motorky a jejich reakce (Cmoto::ApplyForces)

GameRecount_render()

- zkrácené přepočítávání stavu motorky pro rendering (bez kontroly jablek a cíle)
- vyhodnotí stisk kláves spuštěním procedury EvalKeys
- posune a otočí kola motorky o Dt-násobky jednotlivých rychlostí (RecountWheel) a zkontrolují se kolize kol s polygony (WheelCollisions)
- posune a otočí tělo motorky (s jezdcem)
- vyhodnotí síly mezi koly a zbytkem motorky a jejich reakce (Cmoto::ApplyForces)

WheelCollisions()

- detekuje kolize kola s terénem a reaguje na ně
- pokud se zjistí, že během posledního cyklu nastala kolize s terénem (pomocí Cball::collision), vrátí se kolo na místo kolize, nastaví se spočítaná odrazová rychlost a rotace a potom se kolo posune o čas, který zbyl z minulého posunu nedokončeného kvůli kolizi
- to se opakuje, dokud ještě zbývá nějaký čas aktuálního kroku

GameRender()

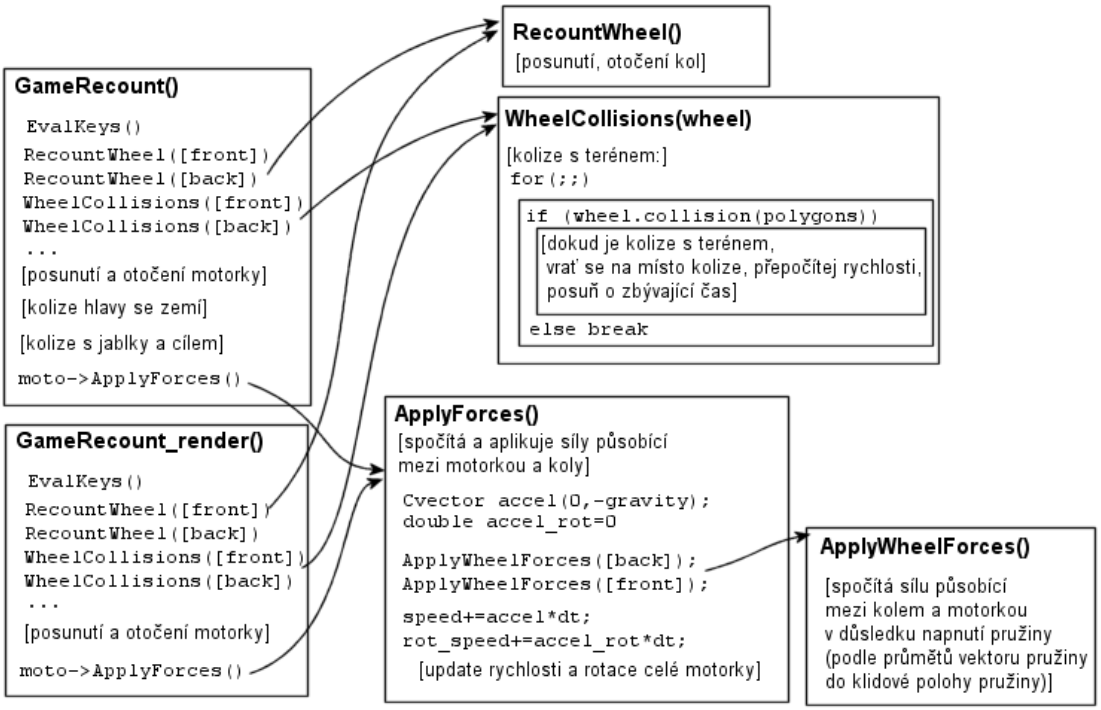
- vykreslí všechny viditelné objekty na obrazovku pomocí funkcí OpenGL: nebe, polygony, kola, pružiny a tělo všech motorek na trati, jablka, cíl a mapu
- před kreslením všech těchto objektů kromě mapy se nastavuje textura metodou bind příslušného objektu Ctexture (viz sekci 6.15)

GameInitVars()

- načte trať a inicializuje všechny proměnné

GameInit()

- spustí GameInitVars, načte textury ze souborů procedurou LoadTextures



Obrázek 6.2: Diagram běhu programu — přepočítávání.

GameDeleteVars()

- smaže a uvolní všechny proměnné

GameEnd()

- spustí GameDeleteVars, smaže textury procedurou DeleteTextures

EvalKeys()

- vyhodnotí stav stisknutých kláves
- pokud se přepočítává jízda ze záznamu, klávesový vstup se přečte ze záznamu
- pokud se přepočítává jízda jiného hráče v multiplayeru, klávesový vstup se získá z objektu hráče (CmultiPlayer) předaného jako parametr funkce
- pokud se přepočítává jízda lokálního hráče, klávesový vstup se získá systémovou funkcí GetAsyncKeyState; pokud se pořizuje záznam jízdy, přidá se stav kláves do záznamu; pokud je aktivní hra více hráčů, uloží se stav kláves do objektu hráče (CmultiPlayer) jako položka keys

6.6 Networking.cpp

- důležité globální proměnné související s hrou více hráčů:
  - multi_players — počet hráčů
  - multiplayer_server — daný hráč je server
  - multiplayer_mode — mód multiplayeru

class CmultiPlayer

- reprezentace hráče ve hře více hráčů
- obsahuje informace o hráči: jméno (nickName), osobní barvu (color), IP adresu jeho počítače (ip), skóre (score), stav naposledy stisknutých kláves (keys), objekt Cmoto jeho motorky (moto), čas posledního přepočítávání motorky (recounts_last) a jeho výsledek (recountResult), čas dokončení trati (recounts_finished), příznaky o tom, jestli je hráč server (server), jestli je to lokální hráč (local), stav připojení (connected) a další
- jedna z položek je také ukazatel na další objekt třídy CmultiPlayer kvůli možnosti vytvoření spojového seznamu hráčů
- metody pro přidání dalšího objektu hráče na konec seznamu (add), smazání daného hráče i všech jeho následníků (DeleteAll), inicializaci objektů motorky všech hráčů v seznamu (initMotoAll), nalezení hráče s daným jménem v seznamu (findNickName)
- ve hře je použita globální proměnná players typu CmultiPlayer*, která reprezentuje seznam všech hráčů (kromě lokálního u serveru)
- zvlášť je ještě uložen ukazatel na objekt lokálního hráče playerLocal, který se používá k předávání lokálních informací ostatním hráčům
- ukazatelé na objekty všech hráčů (včetně lokálního) jsou uloženy také v poli playersInGameList seřazené podle skóre kvůli možnosti zobrazování výsledků ve správném pořadí

class CmultiPlayerClient: public CmultiPlayer

- rozšířená třída hráče v multiplayeru pro uložení potřebných informací o klientech serveru
- mezi rozšiřujícími položkami jsou socket spojení s klientem (sock), handle vlákna komunikujícího s klientem (threadHandle), pozice v seznamu odstraněných jablek (pro mód *Shared apples* nebo *Black Hat & apples*) a fronta údajů k poslání klientovi (CsendDataQ *sendQ) spolu s metodami pro přidávání položek na konec a odeslání této fronty (add_sendQ a send_sendQ)

class CsendDataQ

- fronta dat o ostatních hráčích určená k odeslání klientovi serverem
- každá položka fronty obsahuje ukazatel na další položku seznamu (next), ukazatel na data (data) a typ těchto dat type
- hodnotou položky type je určen skutečný typ dat:
  - add_player — data je ukazatel na objekt třídy dataAddPlayer, který informuje klienta o přidání nového hráče (obsahuje údaje o hráčově jménu, IP adrese a osobní barvě)
  - delete_player — data je ukazatel na objekt třídy dataDelPlayer, který informuje klienta o odebrání hráče s daným pořadím v seznamu
  - konstantu start_timer využívá server k oznámení začátku hry

InitNetwork()

- inicializuje síťovou komunikaci
- pokud funkci volá server, asociuje vytvořený socket s předdefinovaným portem (31072) a začne naslouchat požadavkům na připojení klientů (funkcemi bind a listen)
- funkce volaná na straně klienta naváže spojení k serveru funkcí connect
- vrací vytvořený síťový socket

NetworkClientInit()

- vymění si se serverem základní informace o hře
- server nejprve zkontroluje, že má klient stejnou verzi programu (podle kontrolního součtu — viz sekci 6.2) a že jeho účastí ve hře není překročen maximální počet hráčů
- potom klient obdrží od serveru údaje o trati, jejím kontrolním součtu (pro ověření stejné verze trati), maximálním počtu hráčů, módu multiplayeru a časovém limitu
- následuje odeslání informací o klientovi pro server: klientovo jméno a osobní barva
- nakonec server klientovi pošle informaci o neúspěchu inicializace, pokud klientovo jméno už někdo použil, v opačném případě o jejím úspěchu

NetworkServerInit()

- protějšek funkce NetworkClientInit na straně serveru
- po výměně informací o hře s klientem čeká funkce na zaslání dalšího paketu; klient ho pošle, až když načte trať a je připraven ke hře

BroadcastThread()

- procedura vlákna s handle BroadcastThreadHandle určeného k oznámení klientům ve stejné podsíti o hře konané na daném serveru
- dokud server nezahájí hru, vysílá procedura se sekundovým intervalem všesměrové UDP pakety s informací o názvu trati a módu multiplayeru

NetworkServerAcceptThread()

- vstupní bod vlákna s handle NetworkServerAcceptThreadHandle, které přijímá příchozí připojení klientů k serveru
- nejdřív se zde inicializuje objekt lokálního hráče playerLocal a pokud je zvolen mód multiplayeru *Shared apples* nebo *Black hat & apples*, vytvoří se také pole applesDisabled, aby mohl server informovat klienty o odstraněných jablkách
- poté se čeká na příchozí spojení klientů pomocí funkce accept
- po připojení klienta se pro něj vytvoří nový objekt třídy CmultiPlayerClient a vyplní se jeho údaje funkcí NetworkServerInit
- při nezdaru inicializace se klient odpojí, jinak se zvýší počet hráčů (multi_players)
- nový hráč se přidá do seznamu hráčů players, pomocí metody add_sendQ se připraví informace o ostatních hráčích k odeslání novému hráči a stejným způsobem dá o tomto hráči vědět ostatním
- nakonec se pro nového klienta vytvoří obslužné vlákno s procedurou NetworkServerThread; handle tohoto vlákna se uloží jako položka threadHandle nového hráče
- vnitřek cyklu přijímání klientů je uzavřen do kritické sekce multiplayerStartLock stejně jako zahájení hry v gameMenuClass::process (viz sekci 6.4) kvůli zamezení začátku hry během nedokončeného přijímání klientů

initStartMultiplayer()

- tato funkce je spuštěna po odstartování hry více hráčů: na serveru při zpracování výběru položky start z menu (v gameMenuClass::process), u klienta v síťovém vlákně NetworkClientThread po obdržení povelu k zahájení od serveru
- zastaví přijímání nových hráčů ukončením broadcastového a přijímacího vlákna (BroadcastThread a NetworkServerAcceptThread) a nastartuje samotnou hru
- vytvoří objekty motorky pro všechny hráče (metodou initMotoAll), vytvoří a vyplní pole pořadí hráčů playersInGameList
- inicializuje semafor motoPlayerLockSemaphore (viz 6.3)
- pokud je funkce spuštěna serverem v módu *Black Hat* nebo *Black Hat & apples*, vybere se ještě hráč, který na začátku dostane černý klobouk

NetworkServerThread()

- procedura síťového vlákna serveru — zajišťuje komunikaci s jedním klientem
- jako parametr dostává ukazatel na objekt hráče, kterého má na starosti (CmultiPlayerClient *client)
- ve fázi čekání na zahájení hry informuje klienta o příchodu a odchodu ostatních hráčů: pomocí metody send_sendQ posílá obsah fronty informací o hráčích sendQ klientovi, kdykoliv je tato fronta neprázdná
- po odstartování hry pošle server klientovi příslušnou zprávu spolu s konkrétním údajem o čase do startu
- poté následuje hlavní cyklus předávání údajů o poloze hráčů, jejich skóre a dalších pomocných informací:
    - klient pošle informace o objektu své motorky (položka moto), čase posledního přepočítávání (recounts_last), jeho výsledku (RecountResult), naposledy stisknutých klávesách (keys) a svém skóre (score)
    - kvůli synchronizaci času klient posílá navíc přesnou dobu od posledního přepočtu (viz sekci 5.4)
    - server vrací zpátky spočítanou korekci času, dobu od svého posledního přepočtu, skóre klienta (ve všech módech kromě *Normal mode*, kde si skóre počítá každý hráč sám), případně odebrané jablko (v módech se sdílenými jablky)
    - potom jsou klientovi zaslány údaje (stejné jako od klienta) o ostatních hráčích, a to v pořadí vzniklém přidáváním hráčů (první je vždy server) — toto pořadí se mění pouze podle pokynů serveru, je tedy konzistentní mezi všemi hráči

- uložení informací o hráči do bufferu se provádí funkcí PackMotoData, naopak na jejich čtení se používá UnpackMotoData
- čtení, resp. zápis jednotlivých základních dat se vykonává přetíženými funkcemi bufGet, resp. bufAdd
- za účelem zajištění exkluzivního přístupu ke sdíleným datům lokálního hráče (serveru) se používá semafor motoPlayerLockSemaphore, podobně k datům ostatních hráčů (klientů) se přistupuje v kritické sekci hlídané položkou lock objektu hráče (třídy CmultiPlayer)

NetworkClientThread()

- procedura síťového vlákna klienta — zajišťuje komunikaci se serverem
- protějšek NetworkServerThread() na straně klienta

NetworkFinish()

- ukončuje síťovou komunikaci
- na straně serveru ukončí broadcastové a přijímací vlákno (BroadcastThread a NetworkServerAcceptThread), u klienta zruší síťové vlákno (NetworkClientThread)
- zruší všechny objekty hráčů zavoláním metody deleteAll seznamu všech hráčů (players); u serveru se volá přetížená metoda rozšířené třídy CmultiPlayerClient — to zahrnuje také ukončení vláken všech klientů serveru

playersCheckApplesFinish()

- zjistí dosažení jablek a cíle hráči v módu *Shared apples* (pomocí metody collision položky moto každého hráče)
- kontrolu provádí server, ostatní hráči (klienti) mohou mít zpoždění v přepočítávání, musí se tedy dopočítat pomocí (GameRecount_render)
- pokud nezbývají žádná jablka, zkontroluje se, jestli někdo nedojel do cíle; v takovém případě se mu do skóre připíše bod a hra se ukončí
- pokud na trati ještě zbývají jablka, postupně se zkontroluje jejich dosažení všemi hráči a každému se přidělí tolik bodů, kolika jablek dosáhl
- nakonec se odstraní dosažená jablka metodou disableReached na seznamu jablek apples; zároveň se tato jablka přidají do seznamu odstraněných jablek applesDisabled, aby o nich server mohl informovat klienty, a upraví se počet jablek appleC

playersCheckApplesFinish_no_score_change()

- zjednodušená verze funkce playersCheckApplesFinish určená pro multiplayer mód *Black Hat & apples*
- kontroluje dosažení jablek a cíle, ale nemění skóre hráčů

checkBlackHat()

- zkontroluje předání černého klobouku v módech *Black Hat* a *Black Hat & apples*
- stejně jako v playersCheckApplesFinish se musí dopočítat pozice klientů do aktuálního stavu
- pomocí metody collision položky moto hráče s kloboukem (playerBlackHat) se zjistí kolize tohoto hráče s ostatními
- pokud taková kolize nastala a při minulé kontrole hráč s kloboukem nekolidoval s ostatními (položka blackHat_handover je false), klobouk se předá jinému hráči a tomu se také nastaví blackHat_handover na true
- v opačném případě tato funkce hráči s kloboukem nastaví blackHat_handover na false a umožní mu tak předat klobouk někomu jinému
- nakonec hráči s kloboukem zvýší dobu, po kterou měl klobouk na hlavě (uloženou jako score), o jeden krok (0,01 s)

6.7 EditorWindow.cpp

EditorOpenGLWinMain()

- hlavní procedura editoru — speciálního módu hry pro vytváření tratí
- nejdřív vytvoří okno editoru a nastaví všechny proměnné (EditorInit)
- potom opakuje hlavní cyklus, ve kterém po zpracovávání každé zprávy systému spouští funkci EditorRefresh
- nakonec se uvolní všechny proměnné (EditorEnd)

EditorRefresh()

- přepočítá stav editoru (EditorRecount) a poté smaže obrazovku a překreslí ji (EditorRender)

6.8 Editor.cpp

EditorRecount()

- přepočítá stav editoru: nejdřív zkontroluje stisk tlačítek myši a kláves a zareaguje na ně
  - mění oblast mapy zobrazenou na obrazovce
  - přepíná mezi různými módy editoru (přidávání/editace/posouvání polygonů; přidávání/posouvání jablek; posouvání startu a cíle)
  - při stisku Esc spustí funkci EditorClose (jen pokud byla trať změněna) a podle návratové hodnoty se rozhodne o ukončení editoru

EditorClose()

- uloží trať, když k tomu dá uživatel pokyn prostřednictvím funkce MessageBox
- vrátí true, pokud se má editor ukončit

EditorRender()

- zjednodušeně vykreslí všechny polygony, jablka (aktivní polygon nebo jablko zvýrazní), motorku na startu, cíl, nápovědu
- případně napíše zprávu o neplatném polygonu nebo průniku motorky s polygonem, což předem zjistí

EditorInit()

- načte trať do objektu třídy Ctrack, inicializuje proměnné editoru

EditorEnd()

- smaže a uvolní všechny proměnné použité v editoru

6.9 OpenGLwindow.cpp

- funkce pro operace s hlavním OpenGL oknem
- CreateGLWindow — vytvoření okna; KillGLWindow — zrušení okna
- ShowGLWindow — zobrazení okna
- ResizeWindow — reakce na změnu velikosti okna (změní oblast, kam se vykresluje)

- vytvoření a zrušení rendering a device contextu (SetRenderingContext a KillRenderingContext)
- psaní textu (PrintText), příprava písma na psaní textu (BuildFont)
- SwapBuf — výměna bufferů grafické paměti

OpenGLWndProc()

- procedura okna — zpracovává události systému
- ukládá si stav vstupu uživatele: stisk kláves, tlačítka a pohyb myši
- reakce na WM_PAINT (požadavek na překreslení):
  - v editoru překreslíme obrazovku (EditorRender)
  - ve hře si pouze uložíme příznak na překreslení (proměnná RefreshScreen), jelikož překreslujeme v odděleném vlákně
  - nepoužíváme BeginPaint a EndPaint, protože máme vlastní device context; místo toho ale musíme volat ValidateRect pro označení okna za platné (podobně jako v příkladu v [2] na str. 666)

6.10 Vectors.cpp

class Cvector

- vyjádření 2D vektoru a vektorových operací: má dvě reálné souřadnice, metody pro sčítání, násobení skalárem, skalární součin, otáčení atd.
- základ pro geometrické výpočty ve fyzikální simulaci

6.11 Polygons.cpp

TessellationInit(), TessellationFinish()

- inicializace a ukončení teselace polygonů — polygon se musí rozdělit na konvexní polygony, aby ho funkce OpenGL mohly nakreslit
- používá se knihovna GLU (OpenGL Utility Library)

class Cvertex: public Cvector

- vrchol polygonu jako jeden prvek spojového seznamu všech vrcholů polygonu
- potomek Cvector — souřadnice určují polohu vrcholu v prostoru

class Cpolygon

- polygon — obsahuje začátek spojového seznamu vrcholů (Cvertex)
- metody na vykreslení, přidání vrcholu, posunutí, zjištění křížení hran

class Cpolygons

- polygon jako jeden prvek spojového seznamu polygonů — pro obsažení všech polygonů v trati
- obsahuje jeden objekt Cpolygon jako reprezentaci samotného polygonu

6.12 Balls.cpp

class Cball

- vyjádření kola (disku): obsahuje pozici ve 2D prostoru a rychlost (vektory), úhel otočení, úhlovou rychlost a poloměr

::collision()

- zjišťování kolize s jiným objektem Cball, hranou polygonu, celým polygonem nebo seznamem polygonů (viz sekci 4.3)

::CollisionResponse()

- reakce na náraz s jiným objektem Cball, hranou polygonu nebo vrcholem polygonu — viz obrázek 6.3 nebo sekci 4.3

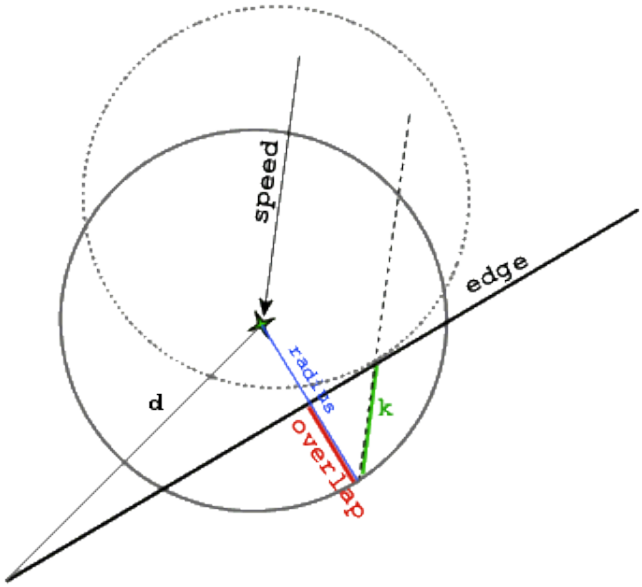
::VertexCollision()

- detekce nárazu s vrcholem polygonu a reakce na něj — viz obrázek 6.4

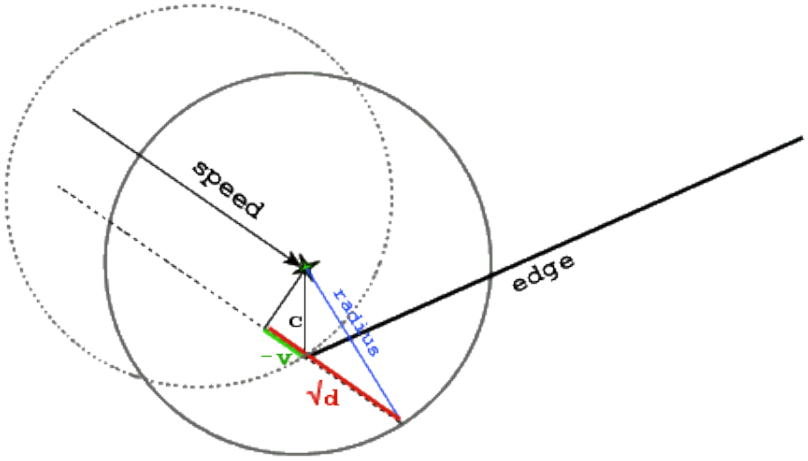
6.13 Motorbike.cpp

class Cmoto

- reprezentace motorky: obsahuje dvě kola (FrontWheel a BackWheel) a hlavu (Head) jako objekty Cball (kvůli snadné detekci kolize); dále pozici, rychlost, úhel a úhlovou rychlost (pos, speed, angle, rot_speed)
- metody pro kreslení motorky a jejích částí, přemístění celé motorky, přepočítání pozice hlavy podle pozice a úhlu motorky (RecountHead), soubor přetížených metod collision pro detekci kolize motorky (kol a hlavy) s polygony, jablky, diskem nebo jiným objektem Cmoto — využívají metody pro příslušnou operaci na jednotlivých částech motorky



Obrázek 6.3: CollisionResponse() — kolize disku s hranou polygonu.



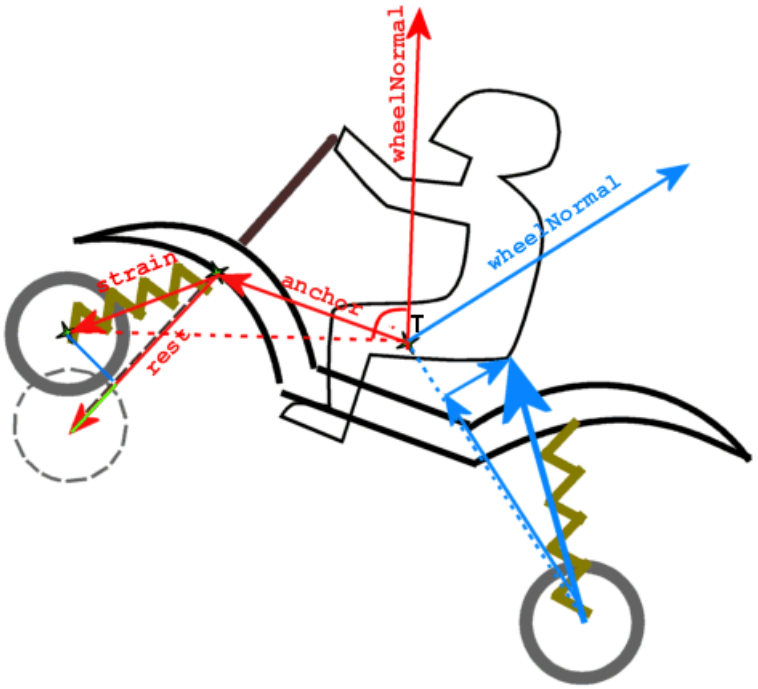
Obrázek 6.4: VertexCollision() — kolize disku s vrcholem polygonu.

::ApplyForces()

- zavoláním ApplyWheelForces na obě kola spočítá síly působící mezi koly a motorkou a aplikuje je na zrychlení jednotlivých kol
- zároveň se opačné síly podle Newtonova zákona akce a reakce (4.4) převedou na zrychlení celé motorky
- zrychlení vynásobené časem od minulého přepočítávání určuje změnu rychlosti motorky, podle které se aktuální rychlost náležitě upraví (podle (4.5))
- analogicky se změní rotační rychlost motorky o rotační zrychlení vynásobené časem
- diagram volání této metody v rámci přepočítávání je na obrázku 6.2

::ApplyWheelForces()

- spočítá síly působící mezi kolem a celou motorkou v důsledku napnutí pružiny a aplikuje je podle (4.5) na změnu rychlostí (posuvné a rotační) kola a motorky
- pro výpočet sil se používá Hookův zákon (vzorec (4.3)), vychýlení pružiny se spočítá pomocí rovinné geometrie vektorů (viz obrázek 6.5)



Obrázek 6.5: Působení sil mezi koly a motorkou.

6.14 Apples.cpp

class Capples

- jeden prvek spojového seznamu jablek
- obsahuje pozici vyjádřenou vektorem — zděděnou od Cvector
- položka enabled určuje, jestli je ještě na mapě nebo již bylo sebráno
- podobně položka reached určuje, jestli již bylo jablko dosaženo (užití v módech multiplayeru se sdílenými jablky — viz sekci 6.6)
- metody pro kreslení (Draw, DrawOne) a pro odstranění dosažených jablek a nastavení reached u odstraněných jablek (disableReached, reachDisabled)

6.15 Texture.cpp

class Ctexture

- třída reprezentující texturu
- metody pro načtení textury ze souboru pomocí knihovny BMGLib (viz [3]), použití textury při kreslení v OpenGL

6.16 Track.cpp

class Ctrack

- obsahuje všechny části trati
- umí je ukládat a načítat ze souboru (metody Load a Save)

6.17 Recorder.cpp

class Crecorder

- třída pro ukládání a přehrávání záznamu hry
- umí záznamy ukládat a načítat ze souboru (metody Load a Save)
- záznamy si ukládá do pole, každá položka je bitové pole stisknutých kláves
- obsahuje metody pro přidávání přehrávání jednotlivých událostí

Literatura

[1] Fauerby K. (2003): Improved Collision detection and Response. http://www.peroxide.dk/papers/collision/collision.pdf

[2] Wright R. S. Jr., Lipchak B. (2005): OpenGL SuperBible, Third Edition. Sams Publishing, Indianapolis, U.S.A.

[3] Bitmapped Graphics Library, http://members.cox.net/scottheiman/bmglib.htm

[4] OpenGL, http://www.opengl.org

Příloha A

Obsah přiloženého CD

Na přiloženém CD se nachází

- uživatelská a programátorská dokumentace ve formátu pdf (tento text)
- zdrojový kód programu
- přeložený spustitelný program s uživatelskou dokumentací
- článek [1]