Karel Klíma
14.5.2010

BouncingBalls – analýza a implementace

1. Popis aplikace

BouncingBalls je program na hraní, který simuluje fyzikální zákony volného pádu a dokonale pružného i nepružného odrazu. Jádrem aplikace jsou balónky a čáry, od kterých se balónky odrážejí. Program také umí produkovat zvukové efekty související s odrazy, které jsou závislé na délce čáry a pozici balónku, takže jde vskutku o „hrací" program.

2. Systémové požadavky

Windows XP nebo Windows Vista nebo Windows 7, 40 MB RAM, Microsoft .NET Framework 3.5.

3. Implementace

3.1. Programovací jazyk a framework

Aplikace je napsaná v programovacím jazyce C# a využívá WPF (Windows Presentation Foundation), jež je součástí Microsoft .NET Framework. Třídy pracující se zvukem jsou pak částečně založeny na open-source knihovně NAudio.

3.2. Důležité objekty a datové struktury

3.2.1. Engine

Třída **`Engine`** představuje výpočetní jádro celého programu. Jejím úkolem je uvádět věci (míčky) do pohybu a koordinovat chování aplikace přesně podle nastavení. Zásadním atributem této třídy je proměnná **`[Int16] Engine.UpdateRate`**, která určuje, kolikrát za sekundu se obnoví obrazovka resp. kolikrát za sekundu se provede výpočet nových pozic všech balónků. Implicitně je nastavená na hodnotu 30, což je dle provedeného měření dobrý kompromis mezi exaktností výpočtu a rychlostí programu.

3.2.2. DraggableControl : Thumb

Tato třída slouží jako základ pro veškeré uživatelsky pohyblivé komponenty. Jediné, co umožňuje, je použití **drag and drop** události, kterou pak využívají třídy **`DropZone`** (představuje místo na obrazovce, ze kterého vypadávají balónky) a **`Anchor`** (představuje jeden ze dvou pohyblivých konců každé čáry).

3.2.3. Ball

Třída **`Ball`** reprezentuje právě jeden balónek. Mezi její nejdůležitější vlastnosti patří proměnná **`[Point] Position`**, která udává aktuální souřadnice, a **`[Vector] Velocity`**, která udává aktuální směr a velikost rychlosti balónku. Balónek je vykreslován pomocí instance třídy **`LineGeometry`**, jejíž vlastnosti (`StartPoint`, `EndPoint`) nastavuje setter proměnné **`Position`**.

3.2.4. Line

Třída **`Line`** reprezentuje pohyblivou čáru. Obsahuje dvě proměnné s instancemi tříd **`Anchor`** (`Anchor1`, `Anchor2`), které tvoří její koncové body. Čára je vykreslována pomocí instance třídy **`LineGeometry`**, jejíž vlastnosti `StartPoint` resp. `EndPoint` jsou nastavovány při drag and drop události objektů `Anchor1` resp. `Anchor2`.

3.3. Implementace fyzikálních jevů

Program využívá pro výpočty pozic balónků kartézský souřadnicový systém a datové struktury **`Point`** a **`Vector`**. Třída Engine pak obsahuje dvě klíčové matematické funkce, které s nimi pracují: 1) funkce **`[Vector] Engine.GetProjection`**`(Vector Vector1, Vector Vector2)`, která spočítá vektorové zobrazení jednoho vektoru do druhého pomocí skalárního součinu; 2) funkce **`[Vector] Engine.GetLeftNormal`**`(Vector Vector)`, která vrací normálu (kolmý vektor) k zadanému vektoru. Veškeré fyzikální výpočty jsou pak prováděny pouze kombinací výše zmíněných funkcí a datových struktur.

3.3.1. Detekce srážky balónku s čárou

Detekce srážky je bezesporu největší problém, se kterým se musí aplikace vypořádat. Během jednoho „kroku" programu se každý balónek může přemístit z místa *A* do místa *B*, kde *A* odpovídá bodu **`Ball.Position`** a *B* odpovídá bodu **`Ball.Position + Ball.Velocity`**. Mezi body *A* a *B* ale může ležet nějaká čára, a to ne nutně pouze jedna. Dále musíme brát v úvahu, že narazí-li balónek na konec (hranu) čáry, musí se zcela jistě odrazit jiným směrem, než pokud narazí na čáru vprostřed – proto se musí kalkulace srážky provádět zvlášť pro konce čar a zvlášť pro hlavní „těla" čar.

Detekce srážky a aplikace případného odrazu je z důvodu výkonu programu zapouzdřena pouze v jedné funkci **`[void] Engine.ApplyPotentialHit`**`(Ball Ball)`. Klíčovým atributem zde proměnná **`[double] Engine.HitDistance`**, která odpovídá kolizní vzdálenosti. Hodnota této proměnné odpovídá poloměru balónků plus poloměru čar.

Algoritmus projde všechny čáry a zjišťuje, zda-li nastala nebo nenastala kolize s balónkem. Následně vybere nejbližší kolizi a aplikuje na ni odraz. Balónek je pak poslán do dalšího kola detekce srážky, protože se musí zjistit, jestli náhodou po změně směru nemůže nastat nějaký jiný odraz – tím se zamezí například nechtěnému prolétávání skrz čáry apod. Cyklus skončí v momentě, kdy není registrována žádná možná srážka balónku s některou z čar.

3.3.2. Aplikace odrazu balónku od čáry

Při výpočtu odrazu se rychlost a směr dopadajícího balónku rozdělí na dvě složky – jedna složka je kolmá na čáru, druhá je s ní naopak rovnoběžná. Při výpočtu odrazové rychlosti přichází na scénu další proměnná, a to **`[double] Engine.BounceAcceleration`**, která odpovídá koeficientu odrazivosti balónku od čáry. Výsledný vektor rychlosti se pak spočítá jako **rovnoběžná složka minus `Engine.BounceAcceleration` krát kolmá složka**.

3.4. Zvuk

Zvukové efekty fungují pouze na 32bitové architektuře. K produkci zvuku je využito nativní Windows WaveOut API, jehož zprostředkování poskytla část knihovny NAudio. V rámci programu se o zpracování zvuku stará třída **`SoundEngine`**.

Odraz balónku od čáry je doprovázen zvukovým efektem v podobě generované sinusové křivky, jejíž frekvence (výška tónu) závisí na délce čáry. Čím kratší čára, tím vyšší tón (a naopak). Frekvence je počítána tak, aby program produkoval pouze existující tóny známé z hudební teorie, nikoliv nějaké nesmyslné pazvuky. Aplikace také implementuje stereo efekt – hlasitost zvuku v jednotlivých výstupních kanálech (levý a pravý reproduktor) je nastavena v závislosti na místě odrazu balónku v okně programu. Jednoduše řečeno – když se balónek odrazí někde hodně vlevo, tak se zvuk ozve spíše z levého reproduktoru (a naopak).