Gymnázium, Praha 6, Arabská 14

Programování

ROČNÍKOVÁ PRÁCE

2024/2025

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

ROČNÍKOVÁ PRÁCE

Předmět: Programování

Téma: Simulace Newtonových fyzikálních zákonů

Autor: Radim Tříletý

Třída: 4.E

Školní rok: 2024/2025

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Prohlášení

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V …….. dne …… ……

Radim Tříletý

Anotace

Toto je dokumentace mé ročníkové práce. V této práci jsou popsány detaily a funkce mého výsledného produktu. Zadáním mého ročníkového projektu bylo vytvořit aplikaci pro simulaci Newtonových zákonů. Začátkem školního roku 2024/25 jsem si vybral toto zadání a snažil jsem se o její zkompletování. Tato aplikace umožňuje uživatelům interaktivně prozkoumat a pochopit první tři Newtonovy pohybové zákony prostřednictvím vizuálních simulací. Uživatel si může vybrat konkrétní zákon a sledovat, jak se objekty chovají v různých situacích. Simulace zobrazují síly působící na tělesa, jejich pohyb a změny rychlosti v závislosti na vnějších silách. Kromě toho aplikace nabízí možnost upravovat parametry, jako jsou hmotnost objektu, velikost síly nebo počáteční rychlost, a tím zkoumat různé scénáře. Pro lepší pochopení principů pohybu obsahuje aplikace i teoretické vysvětlení a příklady k řešení.

Annotation

This is documentation of my year's work. This thesis describes the details and features of my final product. The assignment for my year-long project was to create an application to simulate Newton's laws. At the beginning of the 2024/25 school year, I chose this assignment and worked to make it complete. This app allows users to interactively explore and understand Newton's first three laws of motion through visual simulations. The user can select a specific law and observe how objects behave in different situations. The simulations show the forces acting on the objects, their motion and changes in velocity as a function of external forces. In addition, the application offers the possibility of adjusting parameters such as the mass of the object, the magnitude of the force or the initial velocity to explore different scenarios. For a better understanding of the principles of motion, the application also includes theoretical explanations and examples to solve.

Zadání

Program, který simuluje určité fyzikální jevy, jako je pohyb těles v gravitačním poli (např. Newtonovy zákony pohybu). Cílem by bylo vytvořit program, který simuluje fyzikální pohyb těles s použitím základních zákonů fyziky, zejména Newtonových zákonů. Program by měl umožnit uživatelům vizualizovat trajektorie pohybu objektů (např. kuliček) pod vlivem různých sil (např. gravitace, tření).

Obsah

1 Úvod **1**
2 Newtonovy pohybové zákony **2**
2.1 První Newtonův pohybový zákon (zákon setrvačnosti) 2
2.2 Druhý Newtonův pohybový zákon (zákon síly a zrychlení) 2
2.3 Třetí Newtonův pohybový zákon (zákon akce a reakce) 3
3 Funkčnost aplikace **4**
4 Struktura Aplikace **5**
4.1 Hlavní Menu a Navigace Aplikace 5
4.2 Příklady 6
4.3 Simulace Prvního Newtonova zákon (FirstLawScene.java) 6
4.3.1 updatePhysics 7
4.3.2 Počítání sil působící na objekt 9
4.3.3 Vykreslování sil působící na objekt 10
4.4 Simulace Druhého Newtonova zákon (SecondLawScene.java) 11
4.4.1 Animace pohybu objektu podle druhého Newtonova zákona 11
4.4.2 Mini graf 12
4.5 Simulace Třetího Newtonova zákon (ThirdLawScene.java) 13
4.5.1 Detekce a zpracování kolize koulí 14
4.5.2 Fyzikální síly působící na objekt 15
4.6 Graf k druhému Newtonovu zákonu (SecondLawGraf.java) 16
5 Závěr **18**
Použitá literatura **19**
Seznam obrázků **20**

1 Úvod

Tento dokument slouží jako podrobná dokumentace k ročníkovému projektu zaměřenému na simulaci prvních tří Newtonových pohybových zákonů. Projekt byl vytvořen v jazyce Java s využitím JavaFX, což umožňuje nejen vizualizaci fyzikálních jevů, ale také interaktivní prvky, díky nimž si uživatelé mohou jednotlivé zákony prakticky vyzkoušet. Aplikace kombinuje teoretické vysvětlení se simulacemi, které demonstrují reálné chování objektů při působení různých sil. Hlavním cílem projektu je poskytnout intuitivní a dynamický způsob, jak se studenti i širší veřejnost mohou seznámit s klíčovými principy klasické mechaniky.

V rámci aplikace mohou uživatelé nastavovat parametry jako síla, hmotnost či tření a sledovat, jak se mění pohyb tělesa v závislosti na těchto proměnných. Kromě samotných vizuálních simulací obsahuje aplikace i analytické nástroje, například grafické znázornění rychlosti nebo síly působící na objekt. Díky tomu je možné nejen pozorovat chování objektů, ale také se seznámit s matematickým popisem fyzikálních zákonů. Dokumentace podrobně popisuje strukturu kódu, klíčové funkce aplikace a postupy implementace jednotlivých simulací. Zahrnuje také analýzu problémů, které se v průběhu vývoje vyskytly, spolu s jejich řešením.

2 Newtonovy pohybové zákony

Newtonovy pohybové zákony tvoří základ klasické mechaniky a popisují vztahy mezi silami a pohybem těles. Tyto tři zákony formuloval Isaac Newton v 17. století a dodnes jsou klíčové pro pochopení dynamiky pohybu.

2.1 První Newtonův pohybový zákon (zákon setrvačnosti)

První Newtonův pohybový zákon, známý také jako zákon setrvačnosti, stanovuje, že každé těleso setrvává v klidu nebo v rovnoměrném přímočarém pohybu, pokud na něj nepůsobí vnější síly nebo pokud je výslednice sil nulová. Jinými slovy, těleso samo od sebe nemění svůj pohybový stav, pokud k tomu není donuceno vnější silou.

Tento zákon popisuje setrvačnost tělesa, což je jeho přirozená tendence udržet si svůj pohybový stav. Pokud se těleso již pohybuje, pokračuje v rovnoměrném přímočarém pohybu, dokud na něj nezačne působit síla, která by změnila jeho rychlost nebo směr. Naopak, pokud je těleso v klidu, zůstane v tomto stavu, dokud na něj nepůsobí síla, která jej uvede do pohybu.

V běžném životě lze tento zákon pozorovat například při bruslení na ledě. Když bruslař odrazí nohou a přestane se aktivně pohybovat, stále klouže po ledu díky setrvačnosti, dokud jej nezastaví tření nebo jiná síla. Dalším příkladem je situace v autě – pokud vozidlo náhle zastaví, cestující se vlivem setrvačnosti pohybuje dál dopředu, což je důvod, proč je nutné používat bezpečnostní pásy.

2.2 Druhý Newtonův pohybový zákon (zákon síly a zrychlení)

Druhý Newtonův pohybový zákon popisuje vztah mezi silou, která působí na těleso, a zrychlením, které těleso v důsledku této síly vykazuje. Tento zákon říká, že zrychlení tělesa je přímo úměrné výslednici sil, které na něj působí, a nepřímo úměrné jeho hmotnosti. Matematicky je tento zákon vyjádřen rovnicí:

$$F = m \cdot a$$

kde F je výsledná síla působící na těleso, m je hmotnost tělesa a a je jeho zrychlení.

Podle tohoto zákona je tedy jasné, že čím větší síla působí na těleso, tím větší zrychlení toto těleso vykazuje, pokud je hmotnost tělesa konstantní. Naopak, čím větší hmotnost tělesa, tím menší zrychlení bude vykazovat při působení stejné síly.

Příkladem tohoto zákona může být situace, kdy tlačíme na prázdný vozík a vozík s nákladem. Pokud na oba vozíky působíme stejnou silou, vozík s nákladem se bude zrychlovat pomaleji než prázdný, protože jeho hmotnost je větší. Na druhé straně, pokud bychom na prázdný vozík aplikovali větší sílu, jeho zrychlení by bylo výrazně větší než u vozíku s nákladem.

2.3 Třetí Newtonův pohybový zákon (zákon akce a reakce)

Třetí Newtonův pohybový zákon, známý také jako zákon akce a reakce, říká, že každá akce vyvolá rovnou a opačnou reakci. To znamená, že pokud jedno těleso působí silou na jiné těleso, druhé těleso na první působí silou stejné velikosti, ale opačného směru. Tato síla působí vždy současně, a to i v případě, kdy se tělesa nacházejí v různých vzdálenostech nebo vzorcích pohybu.

Matematicky je tento zákon vyjádřen rovnicí:

$$F = \quad - F_2$$

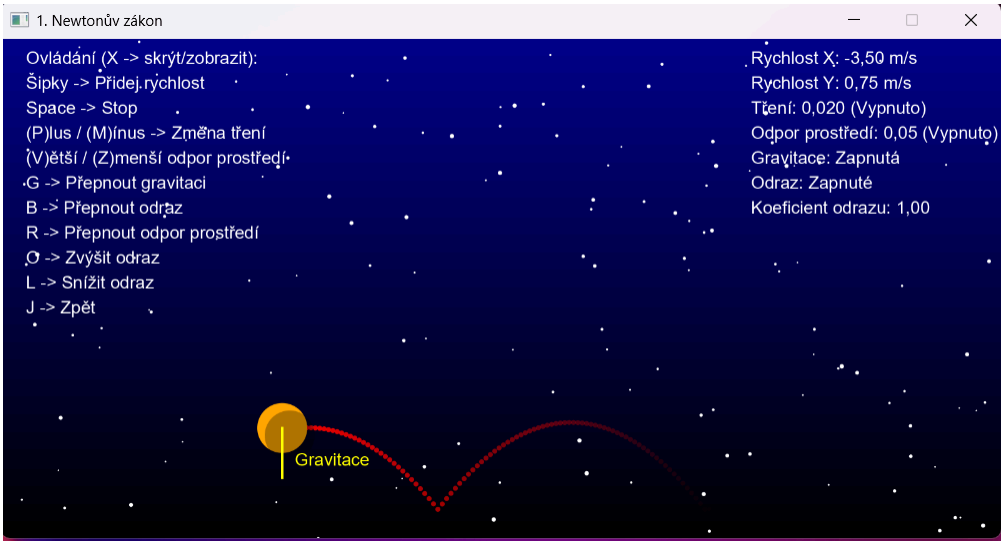
kde F je síla, kterou těleso 1 působí na těleso 2, a F_2 je síla, kterou těleso 2 působí na těleso 1. Tyto síly mají stejnou velikost, ale opačný směr.

Příklady tohoto zákona lze najít v každodenním životě. Například při chůzi chodíme díky tomu, že naše noha tlačí na zem a země tlačí na naši nohu zpět. Pokud bychom vkládali sílu do skákání, tlačíme odrazovou silou na zem, a země nás odrazí zpět. Dalším příkladem je výstřel z pušky, kdy puška působí silou na střelu a střela působí stejnou, ale opačnou silou na pušku, což způsobí zpětný ráz.

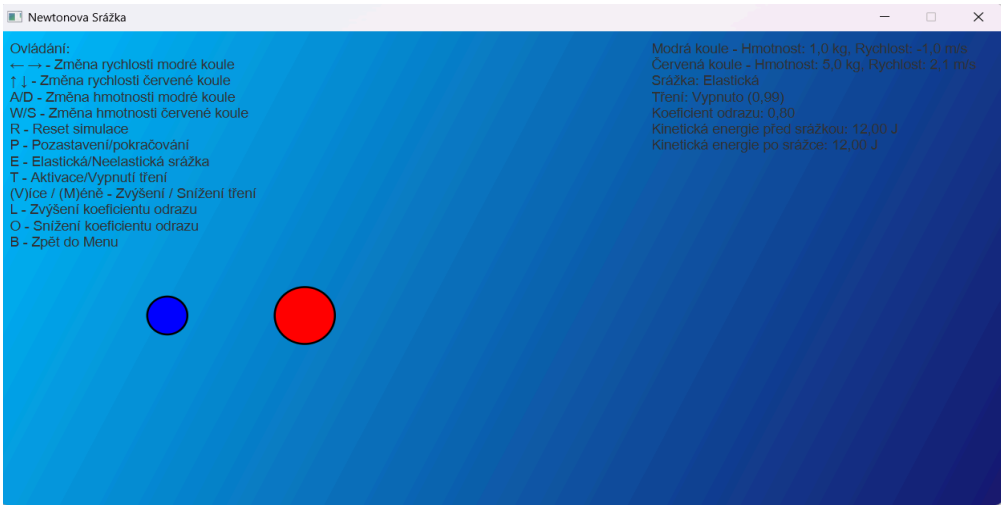
3 Funkčnost aplikace

Aplikace pro simulaci Newtonových zákonů byla navržena s cílem poskytovat uživatelům interaktivní a vizuální prostředí pro lepší pochopení základních fyzikálních principů. Funkčnost aplikace se dělí na několik klíčových částí, které umožňují uživatelům interagovat s fyzikálními simulacemi a experimentovat s různými parametry, které ovlivňují pohyb těles.

V simulacích mohou uživatelé měnit různé parametry jako je rychlost, hmotnost a síla působící na objekty, a sledovat, jak se mění jejich pohyb. Každá simulace obsahuje tlačítka pro změnu těchto parametrů a také pro resetování simulace nebo přepínání mezi elastickými a neelastickými srážkami. Simulace zohledňuje základní principy jako zachování hybnosti a kinetické energie, což je zobrazeno v reálném čase.



obr. 1: Ukázka aplikace 1



obr. 2: Ukázka aplikace 2

4 Struktura Aplikace

Každý Newtonův zákon je implementován jako samostatná třída, která spravuje jak teoretické, tak interaktivní části aplikace. Třídy pro teoretické sekce (například *FirstLawTheory*, *SecondLawTheory*, *ThirdLawTheory*) obsahují textová vysvětlení a tlačítka pro přechod na simulace nebo příklady. Simulace samotné jsou realizovány ve třídách jako *FirstLawScene*, *SecondLawScene* a *ThirdLawScene*, které vykreslují pohyb objektů a poskytují uživatelům možnost interakce. Hlavní struktura zahrnuje:

1. **Hlavní menu** – Tato sekce je výchozím bodem aplikace, kde uživatel vybírá, který z Newtonových zákonů chce studovat. Každý zákon je doprovázen tlačítky pro zobrazení teoretických informací, simulace a příkladů.

2. **Teoretické sekce** – Každý Newtonův zákon má svou vlastní sekci, která vysvětluje klíčové fyzikální principy.

3. **Příklady** – Pro každý zákon jsou připraveny příklady, které pomáhají uživatelům lépe pochopit, jak aplikovat Newtonovy pohybové zákony k řešení konkrétních problémů. Uživatelé se mohou podívat a na grafické znázornění.

4. **Simulace pohybu** – Každý zákon má vlastní interaktivní simulaci, kde si uživatelé mohou upravit parametry jako hmotnost, rychlost nebo sílu působící na objekty a sledovat, jak tyto změny ovlivňují pohyb. Simulace nabízí různé možnosti, jak experimentovat s tělesy.

4.1 Hlavní Menu a Navigace Aplikace

Třída *MainMenuApp* je hlavním rozhraním aplikace a zajišťuje navigaci mezi různými částmi aplikace, která simuluje Newtonovy pohybové zákony. Po spuštění aplikace se uživateli zobrazí hlavní menu s tlačítky pro výběr konkrétního Newtonova zákona. Toto menu obsahuje tlačítka pro každý z Newtonových zákonů a také pro návrat zpět do hlavního menu. Grafické rozhraní je tvořeno pomocí VBox, což je vertikální rozložení, které zajišťuje, že všechny prvky budou uspořádány v jedné vertikální linii.

Každý z Newtonových zákonů má svou vlastní třídu, která se stará o zobrazení zakliknutého zákona a navigaci k dalším funkcím aplikace, jako jsou simulace a příklady. Všechny třídy využívají podobné designové prvky, což zajišťuje konzistentní a přehledné uživatelské rozhraní.

Pozadí**:** V každé třídě je nastaveno pozadí s lineárním gradientem, což dodává aplikaci estetický vzhled a zároveň pomáhá odlišit jednotlivé sekce.

Texty**:** Texty jsou formátovány s různými fonty pro titulky a vysvětlující texty. Titulky mají velký font pro zvýraznění, zatímco vysvětlující texty používají menší písmo pro lepší čitelnost.

Tlačítka**:** Tlačítka jsou vizuálně přitažlivá díky gradovanému pozadí a efektům při najetí kurzorem. Zajišťují snadnou interakci a navigaci mezi jednotlivými částmi aplikace.

4.2 Příklady

V rámci aplikace byly implementovány interaktivní příklady pro každý z prvních tří Newtonových zákonů. Tyto příklady mají za cíl pomoci uživatelům lépe pochopit teoretické aspekty jednotlivých zákonů prostřednictvím praktických situací, které ilustrují chování objektů v reálném světě. Každý zákon je zastoupen samostatnou třídou, která obsahuje konkrétní příklady, uživatelské rozhraní pro výběr odpovědí a metody pro poskytování zpětné vazby.

Třída *FirstLawExamples* se soustředí na první Newtonův zákon. Tato třída obsahuje několik příkladů, které uživatelům ukazují praktické aplikace tohoto zákona v různých scénářích. Třída využívá dvě Mapy (*HashMap*) k uchování dat. Mapa *examples* obsahuje názvy příkladů jako klíče a pole, která obsahují text otázky a tři možnosti odpovědí. Při výběru příkladu uživatel vidí otázku a tři možnosti odpovědí, které se vztahují k dané situaci, jako například, co se stane při vzájemné interakci dvou objektů. Mapa *correctAnswers* obsahuje pro každý příklad správnou odpověď.

Třída *SecondLawExamples* se zaměřuje na druhý Newtonův zákon. Podobně jako třída pro první zákon, třída *SecondLawExamples* používá Mapy (HashMap) k uchování příkladů a správných odpovědí. Uživatelé mohou vybírat příklady a odpovědi z dostupných možností, a to prostřednictvím ComboBoxů.

Třída *ThirdLawExamples* se zaměřuje na třetí Newtonův zákon. Stejně jako v předchozích třídách, Mapy se používají pro uchování příkladů a správných odpovědí.

4.3 Simulace Prvního Newtonova zákon (*FirstLawScene.java*)

Tato část aplikace představuje scénu, která simuluje pohyb objektu podle prvního Newtonova zákona. V této scéně se uživatelé mohou seznámit s tím, jak různé síly ovlivňují pohyb objektu, který je znázorněn jako oranžová koule. Aplikace umožňuje simulovat různé fyzikální efekty, jako je gravitace, tření, odpor prostředí a odraz při nárazu, a to vše s interaktivními ovládacími prvky.

Scéna je vykreslena na plátno pomocí JavaFX, kde je objekt, reprezentovaný koulemi, pohybován na základě různých fyzikálních sil. Pomocí klávesnice mohou uživatelé manipulovat s rychlostí objektu (pomocí šipek nebo klávesy "Space" pro zastavení), aktivovat a deaktivovat jednotlivé síly (gravitaci, tření, odpor prostředí a odraz) a upravit jejich

intenzitu. Tření je modelováno jako síla, která zpomaluje objekt v závislosti na jeho rychlosti, zatímco odpor prostředí působí proti pohybu objektu v závislosti na jeho rychlosti a nastaveném koeficientu odporu. Gravitace působí na objekt tak, že zvyšuje jeho rychlost směrem dolů, pokud je zapnuta.

Aplikace rovněž zobrazuje různé vizuální indikátory pro tyto síly. Například, pokud je aktivována gravitace, na obrazovce se objeví žlutá čára označující směr a velikost gravitační síly. Podobně, pokud je zapnutý odpor prostředí nebo tření, zobrazuje se vizuální znázornění těchto sil, aby uživatelé viděli, jak ovlivňují pohyb objektu. Scéna také zahrnuje trail (stopu pohybu), která ukazuje historii pohybu objektu, což může být užitečné pro vizuální pochopení změn v rychlosti a směru pohybu objektu.

Celkově aplikace kombinuje interaktivitu s fyzikálními simulacemi a dává uživatelům příležitost experimentovat s různými parametry a pozorovat, jak se mění chování objektu v závislosti na změnách těchto parametrů. Tato část aplikace poskytuje vizuálně přehledný způsob, jak pochopit Newtonovy pohybové zákony a jejich aplikace v reálném světě.

4.3.1 updatePhysics

Metoda updatePhysics je klíčová pro simulaci pohybu objektu ve scéně a jeho interakce s různými fyzikálními silami, jako je gravitace, odpor prostředí, tření a odraz.

```
x += velocityX;

if (gravityEnabled) {
    velocityY += 0.15;
}

y += velocityY;

if (ResistanceEnabled) {
    double airResistanceX = -ResistanceCoefficient * velocityX;
    velocityX += airResistanceX;
}
```

obr. 3: Gravitace + Odpor prostředí

Na začátku metody se aktualizuje pozice objektu přičtením aktuální rychlosti ve směru osy X a Y. Pokud je zapnutá gravitace, metoda zvýší rychlost ve směru osy Y o konstantní hodnotu 0.15, což simuluje gravitační zrychlení. Tento přístup umožňuje realistické zrychlování objektu směrem dolů, dokud se nesetká s povrchem.

Pokud je aktivován odpor prostředí, vypočítá se síla odporu působící proti směru pohybu na základě součinitele odporu a aktuální rychlosti objektu. Tato síla je poté aplikována na rychlost objektu, čímž dochází k postupnému zpomalování pohybu, což odpovídá reálnému chování objektů pohybujících se vzduchem.

```
if (y > HEIGHT - 45) {
    y = HEIGHT - 45;
    if (bouncingEnabled) {
        velocityY = -Math.abs(velocityY) * bounceFactor;
    } else {
        velocityY = 0;
    }
}

if (y < 0) {
    y = 0;
    if (bouncingEnabled) {
        velocityY = Math.abs(velocityY) * bounceFactor;
    } else {
        velocityY = 0;
    }
}

if (y >= HEIGHT - 45 && frictionEnabled && friction > 0) {
    double frictionForce = friction * velocityX;
    velocityX -= Math.signum(velocityX) * Math.min(Math.abs(frictionForce), Math.abs(velocityX));
    if (Math.abs(velocityX) < 0.01) velocityX = 0;
}

if (x > WIDTH) x = 0;
if (x < 0) x = WIDTH;
```

obr.4: Kolize scény + Tření

Další část kódu se zabývá kolizí s hranicemi scény. Pokud objekt narazí na spodní okraj scény, jeho pozice se nastaví na pevnou hodnotu, která odpovídá zemi. Pokud je zapnutý režim odrazu, objekt se odrazí zpět se násobený procentem odrazu, což simuluje neelastický odraz. Pokud odraz není aktivován, rychlost objektu se v ose Y nastaví na nulu, což znamená, že objekt zůstane na místě. Podobný princip platí i pro horní hranici scény.

Tření je aplikováno pouze tehdy, pokud se objekt dotýká podlahy a tření je zapnuté. Vypočtená síla tření zpomaluje horizontální pohyb objektu až do úplného zastavení. To simuluje reálné chování těles, která se pohybují po povrchu s třením.

Metoda také zajišťuje, že objekt nemůže opustit scénu v horizontálním směru – pokud se dostane mimo levou nebo pravou hranici, okamžitě se přesune na opačnou stranu, což vytváří efekt nekonečné smyčky.

```
        trail.add(new double[]{x + 20, y + 20});
        if (trail.size() > TRAIL_LENGTH) {
            trail.remove( index: 0);
        }
    }
```

obr. 5: Vizualizace dráhy

Na závěr se přidává aktuální pozice objektu do seznamu *trail*, který uchovává posledních 100 pozic objektu. Tento seznam slouží k vykreslení trajektorie pohybu, což vizuálně znázorňuje dráhu, kterou objekt urazil. Pokud seznam přesáhne nastavenou délku, nejstarší pozice se odstraní, aby byl zachován požadovaný počet bodů.

4.3.2 Počítání sil působící na objekt

Pohyb objektu: $x = x_0 + v_x$; $y = y_0 + v_y$ Tyto rovnice aktualizují polohu objektu v každém snímku na základě jeho aktuální rychlosti. Poloha v ose X se mění podle horizontální rychlosti v_x, zatímco poloha v ose Y se mění podle vertikální rychlosti v_y.

Gravitační síla: $v_y = v_y + g$ Když je aktivována gravitace, přičítá se k vertikální rychlosti malá konstanta g = 0.15. To odpovídá zrychlení volného pádu, kdy gravitace postupně zvyšuje rychlost objektu směrem dolů.

Odpor prostředí (aerodynamický odpor): $F_o = - k \cdot v_x$; $v_x = v_x + F_o$ Síla odporu prostředí je úměrná rychlosti objektu a působí proti směru jeho pohybu. Koeficient odporu k je uživatelsky nastavitelný (*ResistanceCoefficient*). Tato síla postupně zpomaluje objekt, čímž simuluje reálný efekt odporu vzduchu.

Odraz od země: $v_y = |v_y| \cdot bounceFactor$ Pokud je povolen odraz (*bouncingEnabled*), rychlost při dopadu na zem změní svůj směr a zároveň se sníží na *bounceFactor* původní hodnoty, což odpovídá neelastickému odrazu s koeficientem restituce.

Tření (síla smykového tření): $F_t = \mu \cdot v_x$ Smykové tření je úměrné horizontální rychlosti objektu a působí proti směru pohybu. Součinitel tření μ je nastavitelný (*friction*). Síla tření nemůže způsobit změnu směru pohybu – pouze rychlost snižuje až na nulu, pokud je dostatečně velká.

4.3.3 Vykreslování sil působící na objekt

V rámci simulace prvního Newtonova zákona se vizualizují síly působící na objekt. Tento prvek pomáhá uživatelům lépe pochopit vliv různých sil, jako je gravitace, odpor prostředí a tření.

```
if (gravityEnabled) {
    gc.setStroke(Color.YELLOW);
    gc.setLineWidth(2);
    gc.strokeLine( v: x + 20,  v1: y + 20,  v2: x + 20,  v3: y + 60);
    gc.setFill(Color.YELLOW);
    gc.fillText( s: "Gravitace",  v: x + 30,  v1: y + 50);
}

if (ResistanceEnabled) {
    double airResistanceX = -ResistanceCoefficient * velocityX;
    gc.setStroke(Color.RED);
    gc.setLineWidth(2);
    gc.strokeLine( v: x + 20,  v1: y + 20,  v2: x + 20 + airResistanceX * 30,  v3: y + 20);
    gc.setFill(Color.RED);
    gc.fillText( s: "Odpor prostředí",  v: x + 30,  v1: y + 30);
}

if (y >= HEIGHT - 45 && frictionEnabled && friction > 0) {
    double frictionDirection = Math.signum(velocityX) == 0 ? 0 : -Math.signum(velocityX);
    gc.setStroke(Color.GREEN);
    gc.setLineWidth(2);
    gc.strokeLine( v: x + 20,  v1: y + 20,  v2: x + 20 + frictionDirection * 30,  v3: y + 20);
    gc.setFill(Color.GREEN);
    gc.fillText( s: "Tření",  v: x + 30,  v1: y + 10);
}
```

obr. 6: Vykreslování sil

Pokud je gravitace aktivována (*gravityEnabled*), vykreslí se žlutá čára směřující dolů od středu objektu. Vedle čáry se zobrazí text „Gravitace“ jako vizuální popis síly.

Pokud je odpor prostředí zapnutý (*ResistanceEnabled*), vypočítá se síla odporu jako *-ResistanceCoefficient * velocityX*. Červená čára se vykreslí ve směru této síly, znázorňující odpor vzduchu působící proti pohybu.

Pokud je objekt na podlaze (*y >= HEIGHT - 45*), tření je zapnuté a má nenulovou hodnotu, vykreslí se zelená čára opačným směrem k pohybu objektu. Čára reprezentuje sílu tření brzdící objekt.

4.4 Simulace Druhého Newtonova zákon (*SecondLawScene.java*)

Tato část aplikace je zaměřena na simulaci druhého Newtonova zákona pohybu. Uživatel má možnost ovládat hmotnost objektu a sílu, která na něj působí, což ovlivňuje jeho pohyb. Aplikace zobrazuje pohybující se kouli, jejíž rychlost a zrychlení se neustále aktualizují na základě zadaných parametrů. Tento pohyb je simulován pomocí fyzikálních rovnic, kde síla působící na objekt způsobuje zrychlení, které následně mění jeho rychlost a pozici.

Na hlavní obrazovce je zobrazeno několik komponent, které umožňují interaktivní ovládání simulace. Mezi ně patří posuvníky pro úpravu hmotnosti objektu a síly, které na něj působí. Posuvníky umožňují uživateli dynamicky měnit hodnoty hmotnosti a síly, čímž dochází k okamžitému přepočtu pohybu objektu. Kromě posuvníků aplikace zobrazuje aktuální hodnoty rychlosti, zrychlení a času na samostatných popiscích, což poskytuje uživateli přehled o dynamice simulace.

Simulace využívá animaci, která na základě hodnoty síly a hmotnosti aktualizuje pozici koule v reálném čase. Rychlost a zrychlení objektu se počítají zadanými fyzikálními vzorci a výsledky se následně zobrazují na obrazovce. Graf, který je součástí aplikace, zobrazuje průběh rychlosti v závislosti na čase, což umožňuje uživateli sledovat, jak se rychlost mění během simulace.

4.4.1 Animace pohybu objektu podle druhého Newtonova zákona

Jednou z klíčových částí této simulace je metoda *handle(long now)* v třídě *SecondLawScene*, která je součástí *AnimationTimer*. Tato metoda provádí výpočty zrychlení, rychlosti a polohy objektu v reálném čase na základě druhého Newtonova zákona.

```
double deltaTime = (now - lastTime) / 1_000_000000.0;
double acceleration = force / mass;
velocity += acceleration * deltaTime;
position += velocity * deltaTime * SCALE_FACTOR;
```

obr. 7: Simulace pohybu 2.Newtonova zákona

DeltaTime určuje časový krok mezi jednotlivými snímky v sekundách. Zrychlení se počítá podle vztahu: $a = \frac{F}{m}$. Rychlost se upraví podle vztahu $v = v_o + a \cdot \Delta t$. Nová pozice objektu se vypočítá jako $x = x_0 + v \cdot \Delta t \cdot SCALE\_FACTOR$, kde *SCALE_FACTOR* mění pohyb pro vizuální zobrazení.

```
double forceDirectionX = force > 0 ? 1 : -1;
double forceLength = force * SCALE_FACTOR * 0.3;

forceVector.setStartX(position);
forceVector.setStartY(START_Y);
forceVector.setEndX(position + forceLength * forceDirectionX);
forceVector.setEndY(START_Y);

forceLabel.setLayoutX(position + forceLength * forceDirectionX - 25);
forceLabel.setLayoutY(START_Y - 20);
```

obr. 8: Zobrazení síly

Další část metody se zabývá zobrazením vektoru síly. Nejprve se určí směr síly podle jejího znaménka. Délka vektoru se vypočítá jako součin síly a *SCALE_FACTOR*. Poté se nastaví souřadnice počátku a konce vektoru tak, aby vždy vycházel z aktuální polohy objektu a směřoval správným směrem.

```
ball.setCenterX(position);
velocityLabel.setText(String.format("Rychlost: %.2f m/s", velocity));
accelerationLabel.setText(String.format("Zrychlení: %.2f m/s²", acceleration));
timeLabel.setText(String.format("Čas: %.2f s", (now - startTime) / 1_000_000_000.0));

velocitySeries.getData().add(new XYChart.Data<>( x: (now - startTime) / 1_000_000_000.0, velocity));
```

obr. 9: Aktualizace pohybu

Po výpočtech se aktualizují grafické prvky simulace. Poloha koule se nastaví na nově vypočtenou hodnotu a textové popisky zobrazující aktuální rychlost, zrychlení a čas se aktualizují podle nových hodnot. Aby bylo možné sledovat změnu rychlosti v čase, aktuální hodnoty se přidávají do grafu. Na osu X se zaznamenává uplynulý čas od spuštění simulace a na osu Y aktuální rychlost objektu.

4.4.2 Mini graf

Metoda *createChart()* je zodpovědná za vytvoření grafu, který vizualizuje změnu rychlosti objektu v čase. Tento graf je důležitým prvkem simulace, protože umožňuje uživatelům sledovat, jak se rychlost objektu vyvíjí v závislosti na aplikované síle a hmotnosti.

```
private LineChart<Number, Number> createChart() {
    NumberAxis xAxis = new NumberAxis();
    xAxis.setLabel("Čas (s)");

    NumberAxis yAxis = new NumberAxis();
    yAxis.setLabel("Rychlost (m/s)");

    LineChart<Number, Number> lineChart = new LineChart<>(xAxis, yAxis);
    lineChart.setPrefSize( v: 600,  v1: 200);
    lineChart.setLegendVisible(false);

    return lineChart;
}
```

obr. 10: Mini graf

V první části metody se vytváří dvě číselné osy (*NumberAxis*). Osa X reprezentuje čas a je označena popiskem "Čas (s)", zatímco osa Y zobrazuje rychlost a je opatřena popiskem "Rychlost (m/s)". Tyto popisky usnadňují interpretaci dat v grafu.

Následně se vytváří objekt *LineChart<Number, Number>*, který přijímá obě definované osy. Tento graf bude zobrazovat rychlost v průběhu času jako spojitou křivku.

Pro zajištění správného rozložení grafu v uživatelském rozhraní je nastavena jeho preferovaná velikost na 600 × 200 pixelů. To zajistí, že graf bude dobře čitelný a nebude zabírat příliš mnoho místa na obrazovce.

Nakonec se vypíná legenda (*lineChart.setLegendVisible(false)*), protože v této aplikaci není potřeba zobrazit více datových sérií. Graf bude obsahovat pouze jednu křivku představující rychlost objektu, takže legenda by zde byla nadbytečná.

4.5 Simulace Třetího Newtonova zákon (*ThirdLawScene.java*)

Třída *ThirdLawScene* je součástí aplikace, která simuluje Newtonův třetí pohybový zákon, zaměřující se na srážky mezi dvěma koulemi. Tato část aplikace umožňuje uživatelům experimentovat s různými parametry, jako je hmotnost, rychlost, typ srážky (elastická nebo neelastická) a také aktivace tření. Kuličky v aplikaci jsou zobrazeny jako dvě barevné koule (modrá a červená), jejichž pohyb a interakce jsou vizualizovány na animovaném panelu. Uživatelské ovládání je implementováno prostřednictvím klávesových zkratek, které umožňují upravit rychlosti a hmotnosti koulí, změnit typ srážky nebo aktivovat tření.

Simulace umožňuje modelovat elastické a neelastické srážky. Při elastické srážce se kinetická energie zachovává, zatímco při neelastické srážce část kinetické energie ztrácí svůj původní formát. Kromě toho je možné zapnout nebo vypnout tření, které ovlivňuje rychlost koulí během pohybu. Tření je znázorněno pomocí šipek, které ukazují směr a intenzitu síly tření působící na každou kouli. Aplikace rovněž zobrazuje informace o celkové kinetické energii před a po srážce, což umožňuje uživatelům lépe pochopit dynamiku kolizí.

Veškeré nastavení a interakce jsou doplněny o vizuální prvky, které uživatelům pomáhají snadněji sledovat změny a výsledky jejich experimentů. Třída poskytuje realistickou simulaci, kde uživatelé mohou sledovat a analyzovat účinky různých parametrů na srážky a následně je porovnávat. Uživatelé mohou resetovat simulaci, pozastavit ji, nebo přepnout do jiných scén s vysvětlením teoretických principů, což činí aplikaci interaktivní a edukativní.

4.5.1 Detekce a zpracování kolize koulí

Jedním z hlavních prvků simulace třetího Newtonova zákona je správná detekce a zpracování kolize dvou koulí. Tento proces je klíčový pro realistické chování objektů při srážkách a zahrnuje několik kroků, jako je zjištění kolize, oprava překrytí koulí a výpočet jejich nových rychlostí na základě zákona zachování hybnosti a elasticity.

```
double distance = Math.abs(ball1.getCenterX() - ball2.getCenterX());
double combinedRadius = ball1.getRadius() + ball2.getRadius();

if (distance < combinedRadius) {
    double overlap = combinedRadius - distance;
    double nx = (ball2.getCenterX() - ball1.getCenterX()) / distance;
    ball1.setCenterX(ball1.getCenterX() - nx * overlap / 2);
    ball2.setCenterX(ball2.getCenterX() + nx * overlap / 2);
```

obr. 11: Detekce kolizí

Kolize je detekována porovnáním vzdálenosti mezi středy koulí s jejich součtem poloměrů. Pokud je vzdálenost menší než součet poloměrů, znamená to, že se koule překrývají a došlo ke kolizi. Následně se vypočítá velikost překrytí a koule se posunou tak, aby se oddělily a nebyly ve stavu průniku. Tento krok zajišťuje, že se objekty „nelepí“ na sebe a pohyb zůstává realistický.

```
double newV1 = ((m1 - m2) * v1 + 2 * m2 * v2) / (m1 + m2);
double newV2 = ((m2 - m1) * v2 + 2 * m1 * v1) / (m1 + m2);

if (isElastic) {
    v1 = newV1;
    v2 = newV2;
} else {
    v1 = newV1 * restitutionCoefficient;
    v2 = newV2 * restitutionCoefficient;
}
```

obr. 12: Zákon zachování hybnosti + Elasticita

Po korekci polohy koulí se počítají jejich nové rychlosti na základě zákona zachování hybnosti. Vzorce berou v úvahu hmotnosti koulí a jejich původní rychlosti. Pokud je kolize nastavena jako elastická, nově vypočtené rychlosti se přímo aplikují. V případě neelastické kolize se rychlosti dodatečně upraví tak, aby část kinetické energie byla ztracena, což simuluje například náraz míče o podlahu nebo tlumenou srážku dvou těles.

```
if (ball1.getCenterX() < ball2.getCenterX()) {
    ball1.setCenterX(ball2.getCenterX() - combinedRadius);
} else {
    ball1.setCenterX(ball2.getCenterX() + combinedRadius);
}
```

obr. 13: Fail safe

Nakonec je zajištěno, že koule po výpočtu nezůstanou ve stavu překrytí. Pokud by se i po výpočtu rychlostí dostaly do kolizní pozice, jejich poloha se upraví tak, aby mezi nimi byl zachován odpovídající odstup odpovídající jejich poloměrům. Tento krok zabraňuje situaci, kdy by se koule vzájemně „zasekli“.

4.5.2 Fyzikální síly působící na objekt

V kódu je použito několik fyzikálních vzorců pro simulaci různých aspektů pohybu a interakcí mezi objekty.

```
private double calculateKineticEnergy(double mass, double velocity) {
    return 0.5 * mass * velocity * velocity;
}
```

obr. 14: Kinetická energie

Prvním vzorcem je výpočet kinetické energie, která je základní fyzikální veličinou popisující energii pohybu objektu. Tento vzorec je definován jako: $E_k = \frac{1}{2}mv^2$, kde m je

hmotnost objektu a *v* je jeho rychlost. V kódu je tento výpočet implementován ve funkci calculateKineticEnergy, která vrací kinetickou energii objektu na základě jeho hmotnosti a rychlosti. Tento vzorec se používá pro výpočet kinetické energie obou koulí před a po srážce. Kinetická energie je klíčová pro pochopení, jak se energie mezi objekty přenáší nebo ztrácí při jejich interakcích, jako jsou srážky.

```
double newV1 = ((m1 - m2) * v1 + 2 * m2 * v2) / (m1 + m2);
double newV2 = ((m2 - m1) * v2 + 2 * m1 * v1) / (m1 + m2);
```

obr. 15: Zákon zachování hybnosti

Druhým důležitým vzorcem je zachování hybnosti při elastické kolizi, kde se celková hybnost systému před a po srážce nezmění. Tento princip je základní pro popis pohybu při srážkách. Vzorec pro výpočet nových rychlostí v'_1 a v'_2 obou objektů po srážce je odvozen z zákona zachování hybnosti. Tento vzorec určuje nové rychlosti objektů na základě jejich hmotnosti a počátečních rychlostí. Pokud je srážka elastická, tento vzorec se používá k výpočtu nových rychlostí obou koulí po srážce.

```
if (isElastic) {
    v1 = newV1;
    v2 = newV2;
} else {
    v1 = newV1 * restitutionCoefficient;
    v2 = newV2 * restitutionCoefficient;
}
```

obr. 16: Elasticita

Elastická srážka znamená, že kinetická energie se zachovává, což je reflektováno v kódu tím, že po srážce nejsou žádné ztráty energie. Pokud je srážka neelastická, část kinetické energie se ztrácí. Tento efekt je zohledněn v kódu pomocí následujícího zápisu, kde se rychlosti po srážce sníží o koeficient odrazu.

4.6 Graf k druhému Newtonovu zákonu (*SecondLawGraf.java*)

Tato část aplikace zobrazuje grafickou simulaci druhého Newtonova zákona pohybu. Uživatel může zadat hodnoty pro hmotnost a sílu, což následně ovlivní zobrazené grafy, které

ukazují změnu rychlosti a polohy objektu v závislosti na čase. K tomu je použit graf, který zobrazuje:

1. **Rychlost x Čas**: Ukazuje, jak se rychlost objektu mění v průběhu času podle zadané síly a hmotnosti.

2. **Poloha x Čas**: Zobrazuje, jak se mění poloha objektu v čase při působení síly.

Tato simulace byla přidána jako bonus k aplikaci, aby umožnila uživatelům lépe vizualizovat, jak síla ovlivňuje pohyb objektu v souladu s druhým Newtonovým zákonem. Použití knihovny JFreeChart umožňuje snadné vykreslení těchto grafů, zatímco použití časovače a akcí uživatele dává interaktivní dojem.

Tato funkce nabízí jednoduchý, ale efektivní způsob, jak si uživatelé mohou experimentovat s různými hodnotami a vidět okamžité výsledky v grafech.

5 Závěr

Aplikace pro simulaci Newtonových pohybových zákonů byla vytvořena s cílem poskytnout interaktivní a vizuálně přehlednou formu výuky fyzikálních principů. Umožňuje uživatelům experimentovat s různými scénáři a sledovat, jak se objekty chovají v závislosti na zadaných parametrech. Díky spojení teorie a praktických simulací nabízí efektivní způsob, jak si osvojit zákonitosti mechaniky.

Každá část aplikace obsahuje nejen vysvětlení daného zákona, ale také možnost přímého testování v simulovaném prostředí. Uživatelé mohou měnit různé parametry, jako je síla, hmotnost, tření nebo rychlost, a pozorovat, jak se objekty pohybují a reagují na vnější podněty. Tento přístup podporuje aktivní učení a umožňuje lépe pochopit fyzikální vztahy.

Grafické zpracování aplikace bylo navrženo tak, aby bylo přehledné a intuitivní. Jednotlivé scény jsou vizuálně odlišeny, tlačítka a ovládací prvky jsou dobře čitelné a díky použití efektů, jako jsou stíny nebo barevné přechody, je uživatelský zážitek příjemnější. Důraz byl kladen na jednoduchost ovládání, aby aplikace byla přístupná široké škále uživatelů, od studentů až po učitele fyziky.

Velkou výhodou je také možnost simulovat různé situace a sledovat výsledky v reálném čase. To umožňuje nejen lepší pochopení fyzikálních zákonů, ale také podporuje myšlení uživatelů. Experimentování s různými podmínkami přináší interaktivní zážitek, který je efektivnější než pouhé čtení teorie nebo sledování obrázků v učebnicích.

Celkově aplikace splnila svůj účel a nabízí uživatelům názorný a interaktivní pohled na Newtonovy zákony. Spojení teorie, vizualizace a možnosti přímého experimentování umožňuje hlubší pochopení mechanických principů a poskytuje užitečný nástroj pro vzdělávání.

Použitá literatura

Khan Academy. (n.d.). *Newtonovy pohybové zákony*. Retrieved from https://cs.khanacademy.org/science/fyzika-mechanika/x55c156eef0bfca4e:dynamika/x55c156eef0bfca4e:newtonovy-pohybove-zakony/e/newtons-laws

Umíme fakta. (n.d.). *Newtonovy zákony*. Retrieved from https://www.umimefakta.cz/fyzika/book/cviceni-newtonovy-zakony

Wikipedie. (n.d.). *Zákon zachování hybnosti*. Retrieved from https://cs.wikipedia.org/wiki/Zákon_zachování_hybnosti

Reichl, J. (n.d.). *Zákon zachování hybnosti*. Retrieved from http://fyzika.jreichl.com/main.article/view/33-zakon-zachovani-hybnosti

OnlineSchool.cz. (n.d.). *Newtonovy pohybové zákony*. Retrieved from https://onlineschool.cz/fyzika/newtonovy-pohybove-zakony/

CrashCourse. (2016, March 16). *Newton's Laws: Crash Course Physics #5* [Video]. YouTube. https://www.youtube.com/watch?v=CoDtotgpQ9I

ePet.cz. (n.d.). *Co je kinetická a potenciální energie*. Retrieved from https://www.epet.cz/co-je-kineticka-a-potencialni-energie/

Oracle. (n.d.). *DropShadow (JavaFX 2 API)*. Retrieved from https://docs.oracle.com/javafx/2/api/javafx/scene/effect/DropShadow.html

Stack Overflow. (2020, June 10). *JavaFX AnimationTimer not stopping properly*. Retrieved from https://stackoverflow.com/questions/65061944/javafx-animationtimer-not-stopping-properly

Index.dev. (2021, December 7). *Build a Java GUI Application with JavaFX*. Retrieved from https://www.index.dev/blog/build-java-gui-application-javafx

Oracle. (n.d.). *GraphicsContext (JavaFX 8 API)*. Retrieved from https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/GraphicsContext.html

OpenAI. (n.d.). *ChatGPT*. Retrieved from https://chat.openai.com

Stack Overflow. (2015, March 9). *How to resize a canvas on JavaFX to fit the size*. Retrieved from https://stackoverflow.com/questions/29906483/how-to-resize-a-canvas-on-javafx-to-fit-the-size

Stack Overflow. (2023, December 12). *Change enemy movement upon collision*. Retrieved from https://stackoverflow.com/questions/77634359/change-enemy-movement-upon-collision

Oracle. (n.d.). *EventHandler (JavaFX 15 API)*. Retrieved from https://openjfx.io/javadoc/15/javafx.base/javafx/event/class-use/EventHandler.html

Stack Overflow. (2013, March 22). *Checking collision of shapes with JavaFX*. Retrieved from https://stackoverflow.com/questions/15013913/checking-collision-of-shapes-with-javafx

Stack Overflow. (2019, September 11). *How to add JFreeChart library to JDK*. Retrieved from https://stackoverflow.com/questions/58084891/how-to-add-jfreechart-library-to-jdk-error-package-org-jfree-chart-does-not-ex

JFree. (n.d.). *ChartFactory (JFreeChart API)*. Retrieved from https://www.jfree.org/jfreechart/api/javadoc/org/jfree/chart/ChartFactory.html

JFree. (n.d.). *JFreeChart Forum – Viewtopic*. Retrieved from https://www.jfree.org/forum/viewtopic.php?p=50608

Seznam obrázků

1. Ukázka aplikace 1
2. Ukázka aplikace 2
3. Gravitace + Odpor prostředí
4. Kolize scény + Tření
5. Vizualizace dráhy
6. Vykreslování sil
7. Simulace pohybu 2.Newtonova zákona
8. Zobrazení síly
9. Aktualizace pohybu
10. Mini graf
11. Detekce kolizí
12. Zákon zachování hybnosti + Elasticita
13. Fail safe
14. Kinetická energie
15. Zákon zachování hybnosti
16. Elasticita