ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA PEDAGOGICKÁ

KATEDRA MATEMATIKY, FYZIKY A TECHNICKÉ VÝCHOVY

DIPLOMOVÁ PRÁCE

Plzeň, 2022 **Bc. Josef Rada**

FAKULTA PEDAGOGICKÁ

Studijní program: Učitelství pro základní školy

Bc. Josef Rada

Studijní obor: Učitelství matematiky pro základní školy a Učitelství fyziky pro základní školy

POČÍTAČOVÉ MODELY VYBRANÝCH ÚLOH Z MECHANIKY

Diplomová práce

Vedoucí práce: PhDr. Pavel Masopust, Ph.D.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a všechny použité prameny jsem uvedl v seznamu použitých zdrojů.

V Plzni dne 28. 6. 2022

……

vlastnoruční podpis

Bibliografická identifikace

Příjmení a jméno:	Bc. Josef Rada
Katedra:	Matematiky, fyziky a technické výchovy
Název práce:	Počítačové modely vybraných úloh z mechaniky
Vedoucí práce:	PhDr. Pavel Masopust, Ph.D.
Počet stran – číslované:	70
Počet stran:	75
Počet příloh:	3
Klíčová slova:	fyzika, mechanika, počítačový model, modelování, Unity, C# Script, příkaz, slovní úloha o pohybu, pružná srážka, nepružná srážka

Bibliographical identification:

Surname and name:	Bc. Josef Rada
Department:	Mathematics, physics and technical education
Title of thesis:	Computer models of chosen exercises from mechanics
Consultant:	PhDr. Pavel Masopust, Ph.D.
Number of pages – numbered:	70
Number of pages:	75
Number of appendices:	3
Keywords:	physics, mechanics, computer model, modeling, Unity, C# Script, command, word problem about movement, elastic collision, inelastic collision

Poděkování

Děkuji panu PhDr. Pavlu Masopustovi, Ph.D. za odborné vedení práce, věcné připomínky, vstřícnost při konzultacích, ochotu a podporu, kterou mi v průběhu zpracovávání této práce poskytoval.

Obsah

Úvod

Diplomová práce se zabývá problematikou tvorby počítačových modelů na vybrané úlohy z mechaniky. Toto téma bylo zvoleno z toho důvodu, že je velmi praktické a velmi využitelné ve výuce nejen přírodovědných předmětů. Názornost zejména v přírodovědných předmětech je velmi důležitá, obzvlášť pak v matematice a fyzice. Na trhu neexistuje příliš velké množství programů, které by se při výuce fyziky využívaly. Cílem práce bylo proto vytvořit animace na vybrané úlohy zaměřené na určitou oblast fyziky, které by byly využitelné při výuce k lepšímu znázornění úloh.

Abychom čtenáři přiblížili, čemu se mechanika věnuje a jak se mechanika dostala do popředí, popsali jsme v úvodní části tuto fyzikální oblast. V první kapitole jsou tedy zmíněni někteří významní představitelé, kteří se mechanikou zabývali v jednotlivých etapách jejího rozvoje. Otázkou zůstává, o co všechno bychom bez těchto osobností přišli. Zejména kvantová mechanika, která vycházela z té klasické, totiž hrála významnou roli i při tvorbě mobilů a počítačů.

Díky počítačům jsme schopni vytvářet simulace, které mohou být použity v nejrůznějších oblastech. Jednou z těchto oblastí je školství, ve kterém je názornost zejména v abstraktních vědách velice důležitá. Čtenář je seznámen s postupnou tvorbou modelů a jejich typem. Je důležité, ve které aplikaci jsou modely vytvářeny. Scény jednotlivých programů jsou odlišné strukturou. Většina programů se neshoduje také v programovacím jazyce, kterého využívá pro popis jednotlivých příkazů.

Pro vytváření programů byla zvolena platforma Unity, která k dispozici na trhu od roku 2005. Tato platforma je vcelku přehledná pro vytváření animací. Pracuje s herními objekty, které se umisťují na scénu společně s komponentami. Poté se „volají“ do scriptu, který se musí „pověsit“ na tyto objekty. Tento script využívá programovacího jazyka C#. Vytváření aplikací bylo hlavním cílem této diplomové práce. Ke splnění vytyčeného cíle bylo zapotřebí vybrat úlohy, které budou v této platformě modelovány.

Úlohy z mechaniky byly vybrány na základě subjektivního pocitu, který byl zaměřený na problémové situace studentů základní a střední školy v této oblasti fyziky. Před samotnou modelací bylo popsáno řešení a dopočítána výsledná hodnota, kterou budeme moct porovnat se zjištěnou hodnotou prostřednictvím aplikace.

Vedlejším cílem bylo otestování vytvořených modelů ve výuce fyziky. Bylo provedeno výzkumné šetření dotazníkovou formou. Žáci posuzovali, jestli jim aplikace pomohla k pochopení zadání úlohy, popřípadě k řešení samotné úlohy. To, jestli jim animace pomohla k řešení úlohy, jsme ověřili opětovným zadáním úlohy žákům v následující hodině. Získané výsledky jsou uvedeny v závěru diplomové práce. V přílohách jsou pak zdrojové kódy, které jsou stěžejní součástí chodu animací.

1 Mechanika

1.1 Fyzika

Fyzika je vědou zkoumající přírodu. Označení pochází z řeckého slova „physikos“, což znamená „přírodní“. Fyzika byla původně jedinou vědou, která přírodu zkoumala, v dnešní době se jí zabývá chemie, biologie nebo také geologie, mineralogie a jiné. Jednotlivé vědy jsou ale zaměřeny na konkrétní oblast přírody, kterou zkoumají. V některých situacích nicméně není jednoduché rozhodnout, do jakého odvětví daný jev zařadit, proto jsou i vědy, které vznikly kombinací přírodních věd například biofyzika, astrofyzika a další. Až v 19. století se začal používat název fyzika ve stejném smyslu, ve kterém se používá v dnešní době.

Fyzika se samozřejmě, stejně tak jako ostatní vědy, v průběhu dějin vyvíjela. Poznávání nejen ve fyzice neustále pokračuje a je snahou postupovat vpřed i přes spoustu nezdarů. Historii této vědy lze rozdělit různě v závislosti na jejím počátku. Štoll (2009, str. 18-19) ve své publikaci rozděluje dějiny podle dvou symbolických roků. Prvním z nich je rok 1600, kterým začíná vědecká revoluce a druhým je rok 1900, protože v této době vznikala a byla objasňována kvantová a relativistická fyzika. Na základě toho lze označit období před rokem 1600 jako „starou fyziku“, která zahrnuje především mechaniku (konkrétněji statiku a hydrostatiku) a optiku. Období mezi zmíněnými symbolickými roky je označováno jako klasická fyzika a v tomto období je vnímána jako plnohodnotná věda, jelikož je využíváno přístrojů, měřicích metod a matematických aparátů mezi které se řadí i matematická analýza. V této době se vedle mechaniky a optiky rozvíjela především elektřina a magnetismus, termika, termodynamika a také statistická fyzika. Období po roce 1900 je pak nazýváno moderní fyzikou. Teorie kvantové a relativistické fyziky dokázala zobecnit model klasické mechaniky a přinesla nové představy týkající se částic, které nelze zaznamenat lidskými smysly. (Štoll, 2009, str. 13-20)

1.2 Mechanika jako věda

Mechanika je jedním z oborů fyziky, zabývá se studiem zákonitostí týkajících se pohybu těles. V této oblasti vznikaly základní fyzikální pojmy, které byly ověřovány a objasňovány pomocí matematických modelů. Jedná se o velmi rozsáhlé téma, jehož zkoumání není dosud ukončené.

Její počátky jako vědy jsou dílem významných badatelů z přelomu středověku a novověku, mezi které jsou právem řazeni například Koperník, Kepler, Tycho de Brahe nebo Galileo, které fascinovaly zákony, jimiž se řídila tělesa na Zemi i mimo ni. Neměli to ale v této době vůbec snadné, jelikož církev moc dobře věděla, že její budoucnost byla v ohrožení. Vědecké poznání probíhalo především v Evropě a bylo potřeba rozhodnout, jestli umožnit vědě rozvoj. To se nakonec podařilo právě v počátcích novověku a svobodný rozvoj byl umožněn. (Horský, 2001, str.9; Höschl, 2009, str. 9; Samek, 2014, str. 42; Štoll, 2009, str.106)

1.3 Mechanika ve středověku

Fyzika byla ve středověku orientována především na mechaniku, která dále zkoumala a rozvíjela tři oblasti. První navazovala na Archimedovu práci a Araby a řešila otázku ohledně hmotných těles v relativním klidu a rovnováhy těles. Druhá část korigovala Aristotelovy představy týkající se různých druhů pohybů a třetí část se věnuje příčinám, které způsobují vlastní pohyb a vlastnosti, které pohyb ovlivňují. Nicméně až do 17. století nebyla zodpovězena ani jednoduchá otázka týkající se volného pádu, nicméně je třeba si uvědomit, že do té doby nebyly známy některé fyzikální veličiny, neexistovala řada měřicích přístrojů a matematika nepoužívala funkční závislosti. (Štoll, 2009, str. 114-117)

1.4 Mechanika v renesanci

Evropské období, které se nachází mezi středověkem a novověkem, je označováno jako renesance. V této době byla obnovena nejrůznější díla z antického období a otevřela prostor také fyzice. Mechanika byla rozvíjena především prostřednictvím všeuměla Leonarda da Vinci, který se zabýval nepohybujícími i pohybujícími se tělesy, inspiraci pak hledal především v přírodě. Na da Vinciho práci navazovali zejména fyzikové v Itálii a Holandsku, kterým se podařilo objasnit několik problémů, na jejichž vrcholu bylo odmítání perpetua mobile. Tento poznatek byl velmi důležitý pro pozdější formulování zákonů mechaniky a také jiných oblastí fyziky. (Štoll, 2009, str. 127-134)

1.5 Klasická fyzika

V první polovině 17. století se začaly rozvíjet měřicí přístroje, které umožnily značný pokrok nejen v oblasti mechaniky. Téměř v každém roce docházelo ke zdokonalování těchto přístrojů nebo technologií, čímž se zpřesňovaly výsledky. Objevy v této době se neobešly beze sporů, často byly prezentovány ústně bez přesného důkazu.

Mnoho fyziků mělo v této době významný vliv na rozvoj různých oblastí fyziky, nicméně mezi nejvýraznější osobnost v oblasti mechaniky se řadí Isaac Newton. Byl to právě on, kdo dokázal zformulovat zákony popisující pohyb, které podpořil matematikou. Právě matematika pomohla Newtonovi v popisu problémů, na které se zaměřil. Newtonova mechanika je postavená na tom, že prostor a čas jsou absolutní, což znamená, že prostor je nehybný a čas utíká stále stejně a není závislý na jiných okolnostech.

Dnes už víme, že tyto zákony neplatí pro velmi rychlá tělesa a pro přesný výpočet je důležité použít relativistickou mechaniku. Ještě větším problémem ale je, že pohybový zákon reaguje bez prodlevy na změnu hmoty zejména pro velké vzdálenosti. Lze tedy konstatovat, že Newtonova mechanika se používá dodnes, ale nedokáže objasnit všechny problémy reálného světa.

Nicméně Newtonova mechanika hraje v dnešní fyzice stále významnou roli, slavnou se stala potvrzováním platností pro pohyb vesmírných těles na obloze. Vypadalo to tak, že vše půjde popsat právě pomocí této teorie. Prvním výraznějším úspěchem byla předpověď týkající se pohybu Heliovy komety, a to v roce 1757 nebo 1758. Předpověď byla relativně přesná (kometa byla spatřena v polovině března roku 1759) a klasická mechanika slavila úspěch. (Kulhánek, 2019, str. 15-17; Štoll, 2009, str. 142-250)

1.6 Moderní fyzika

Koncem 19. století se zdálo, že pomocí Newtonovské mechaniky, popisu elektromagnetismu od Maxwella a Maxwellovo-Bolzmannovo statistickou termodynamikou lze popsat všechny jevy. Vědci zpřesňovali dosud získané poznatky využitím matematických modelů. Jenže právě tím se dostávali u několika jevů k rozporům s klasickou fyzikou a otevírali tak brány moderní fyzice. Jedním z příkladů může být neschopnost vysvětlit povahu rentgenových paprsků, u kterých je zapotřebí použít abstraktních pojmů, čemuž nemohl Röntgen uvěřit. Další nejasnosti se týkaly otázky pohybu Země v pružném prostředí nebo rozdělení energie v podání Maxwella a Bolzmana. Těm se nedařilo vysvětlit chování absolutně černého tělesa a teoretické poznatky stále neodpovídaly těm naměřeným. (Herneck, 1974, str.58)

Problémů, které nedokázala klasická mechanika popsat, se objevovalo čím dál více. Za úplně největší problém bylo považováno vyzařování spektrálních čar atomy.

Tyto problémy vzbuzovaly koncem 19. století velké rozpaky a spousta fyziků nedokázala tyto jevy přijmout. Toto období bývá často označováno za krizi fyziky. Krize by ale znamenala bezvýchodnou situaci, avšak v této době probíhaly experimenty, rozvíjela se mezinárodní spolupráce a vznikala nová centra. Málokdo si uměl představit, že fyziku potká obrovský převrat ve vnímání dosud platných zákonů.

1.6.1 Stará kvantová mechanika

Fyziku popisující dnešní svět lze rozdělit na dvě časové etapy. Nástupce klasické fyziky lze nazvat starou kvantovou mechanikou, která se vyvíjela přibližně od roku 1900 do roku 1925. Max Planck na konci roku 1900 vystoupil na zasedání v Berlíně, při kterém představil závislost hustoty energie záření absolutně černého tělesa. Jeho poznatky odpovídaly experimentálně naměřeným hodnotám za předpokladu, že energie je vyzařovaná v malých porcích, tzv. kvantech, která jsou úměrná frekvenci. Byl to první objev, který byl v rozporu s klasickou mechanikou a není tak divu, že spousta fyziků pochybovala o její správnosti. Přesto existovali někteří fyzikové a především chemici, kteří spatřovali v Planckově závěru potvrzení atomismu.

V roce 1905 se k Planckovi přidal Albert Einstein, jedna z nejznámějších osobností fyziky. Právě díky nim je uznávána kvantová fyzika. Byla to právě speciální teorie relativity, která v podstatě zobecňuje klasickou fyziku tím, že dokáže správně popsat pohyby těles a částic pro rychlosti velmi blízké světelné rychlosti. Obsahuje asi nejznámější vzorec celé fyziky
$E = m \cdot c^2$. Tato teorie není snadno pochopitelná, změnila pohled na vnímání času a dnes je experimentálně ověřena. Není pochyb o tom, jak důležitá pro popis fyzikálních jevů je. Využívá se například při konstrukci fyzikálních zařízení.

V následujících letech se Einstein zaobíral tzv. principem ekvivalence, který popisoval padání těles v gravitačním poli a který je základem obecné teorie relativity. V roce 1915 se mu podařilo opravit Newtonův gravitační zákon, za což se výslovně omluvil. Právě provádění oprav klasické mechaniky pomocí kvantování dokázalo vysvětlit některé mikroskopické jevy a stálo tak u zrodu nové fyzikální teorie, která je nazývána moderní kvantovou mechanikou. (Štoll, 2009, str. 383-472)

1.6.2 Moderní kvantová mechanika

Začátkem 20. let 20. století začala tedy vznikat na základě nových myšlenek nová teorie, která měla řešit jevy v atomech. Navazovala na méně přesné teorie, mezi které se řadí kvantová teorie světla nebo teorie stavby atomů zkoumaná N. Bohrem. Na moderní kvantovou mechaniku šlo v té době nahlížet dvěma různými pohledy. Jednalo se o Heiseinbergovu maticovou mechaniku a Schrödingerovu vlnovou mechaniku, o nichž dnes víme, že to jsou pouze různé formy jednoho a toho samého způsobu řešení problémů kvantové mechaniky. Tyto myšlenky umožnily rozvoj relativistického pojetí světa mikročástic a díky nim se při řešení nemusela omezovat jejich rychlost.

Za zakladatele řešení relativistických problémů kvantové mechaniky jsou považováni Dirac a Fock. Jejich poznatky byly využity při vzniku kvantové elektrodynamiky a kvantové teorie pole počátkem 40. let 20. století. Relativistická kvantová mechanika byla využita za poslední půl století především v technice a elektronice. Díky tomu vznikly i tranzistory, které jsou součástí moderních zařízení, mezi které se řadí například mobily a počítače. Kvantová mechanika umožnila také vývoj laseru. Zůstává otázkou, jak byl vypadal a fungoval dnešní svět bez kvantové mechaniky, na které je obrovsky závislý. (Opatrný, 2012, str. 37-38)

2 Počítačové modely

2.1 Model, modelování

V posledních letech se informační technologie stále více zařazují do výuky ve školách, a ne jinak je tomu při výuce fyziky. S využitím počítačů a jiných prostředků lze ve školách zefektivnit nejen práci učitelů, ale především žáků. Nedílnou součástí fyziky jsou experimenty, pozorování nebo práce s modely, které potvrzují předpoklady vycházející z teorie. Jsou to právě většinou matematické modely, na kterých je fyzika nebo jiná věda závislá. (Válek, 2014, str. 22)

Pod pojmem model se rozumí zjednodušení původního, složitého systému jiným systémem, který má určité vlastnosti shodné s vlastnostmi v původním systému. Model často obsahuje matematickou rovnici, která popisuje danou situaci, například vzorce Newtonovy mechaniky, relativistické kvantové mechaniky, aj. Modelování je pak proces, který využívá těchto modelů ke zkoumání vybraných vlastností. Nedílnou součástí řešení je většinou výpočetní technika, která umožní zrychlení matematických výpočtů.

Právě numerické řešení je nejčastěji používanou metodou, zejména pak pro řešení diferenciálních rovnic. Tyto modely a jejich řešení pomáhá žákům v poznávání, dokáže častokrát znázornit a tím přiblížit studentům zkoumané problémy. (Mechlová, 2014; Vachek, 1980)

2.2 Počítačové modely ve fyzice

Právě počítačové modely a animace pomáhají žákům vizualizovat danou situaci, čímž mohou umožnit snazší pochopení vybraného fyzikálního jevu. Žáci tak nemají možnost utvořit si mylnou představu. Pro samotné učení žáků nemusí být samotné modelování vždy výhodou, žáci nemohou opravovat mylnou představu, kterou daný jev vnímají. Bylo by proto vhodnější modely využívat jiným způsobem, který u žáků podporuje jejich prvotní pohled na daný jev. (Válek, 2014, str. 29)

Jedním z prvních představitelů metody ověřování vybraných vlastností, zkoumání nových fyzikálních poznatků nebo chování těles a látek, byl známý teoretický fyzik Richard Phillips Feynman. Na svých přednáškách ale v té době nevyužíval počítače, nýbrž studenty, kteří prováděli výpočty ručně. (Feynman, 2013, str. 127–135)

V současnosti se tato metoda stále využívá při řešení změn výsledků vybraných hodnot v závislosti na změně vstupních parametrů. Žáci se řešením pomocí modelů setkávají s programováním a musí využívat logiky, aby dokázali posoudit, jak moc se změní výsledné řešení se změnou vstupních parametrů. V některých úlohách už si nelze danou problematiku představit, žáci se tedy seznamují především s modely, které ve vědě hrají významnou roli právě v této oblasti.

Příkladem může být změna parametrů při šikmém vrhu. Tradiční by mohla být změna počáteční rychlosti nebo úhlu, pod kterým je těleso vrženo. Sledovaná veličina by byla délka tohoto vrhu. Při úpravě zmíněných parametrů mohou mít někteří žáci problém si uvědomit, jakým způsobem se délka tohoto vrhu změní Počítačové modelování umožňuje změnu také tíhového zrychlení. Realita je trochu jiná, trajektorie je ještě měněna odporem vzduchu.

Ne vždy se ale tyto modely používají k představení fyzikálního jevu, který má být žáky ověřen. Pokud model dokáže dynamicky reagovat na změnu vstupních parametrů, jakými jsou například konstanty popisující tělesa a prostředí, nazývá se dynamickým.

Pro takový model je stěžejní znalost rovnice, která popisuje vybraný jev a samozřejmě i počátečních podmínek. Při znalosti takové rovnice pak stačí zvolit jednu veličinu a sledovat závislost výsledné hodnoty na změně vstupní hodnoty. Při využití těchto modelů je ale potřeba, aby se tvůrce orientoval jak ve fyzice, tak matematice.

Počítačové modely se využívají zejména pro rychlost výpočtů pomocí počítače, které disponují také rychlým zpracováním dat a vykreslení závislosti sledovaných veličin. To umožňuje využití řádově statisíců elementárních kroků v relativně krátkém časovém úseku. (Lepil, 2007, s. 7-67)

2.3 Tvorba počítačového modelu

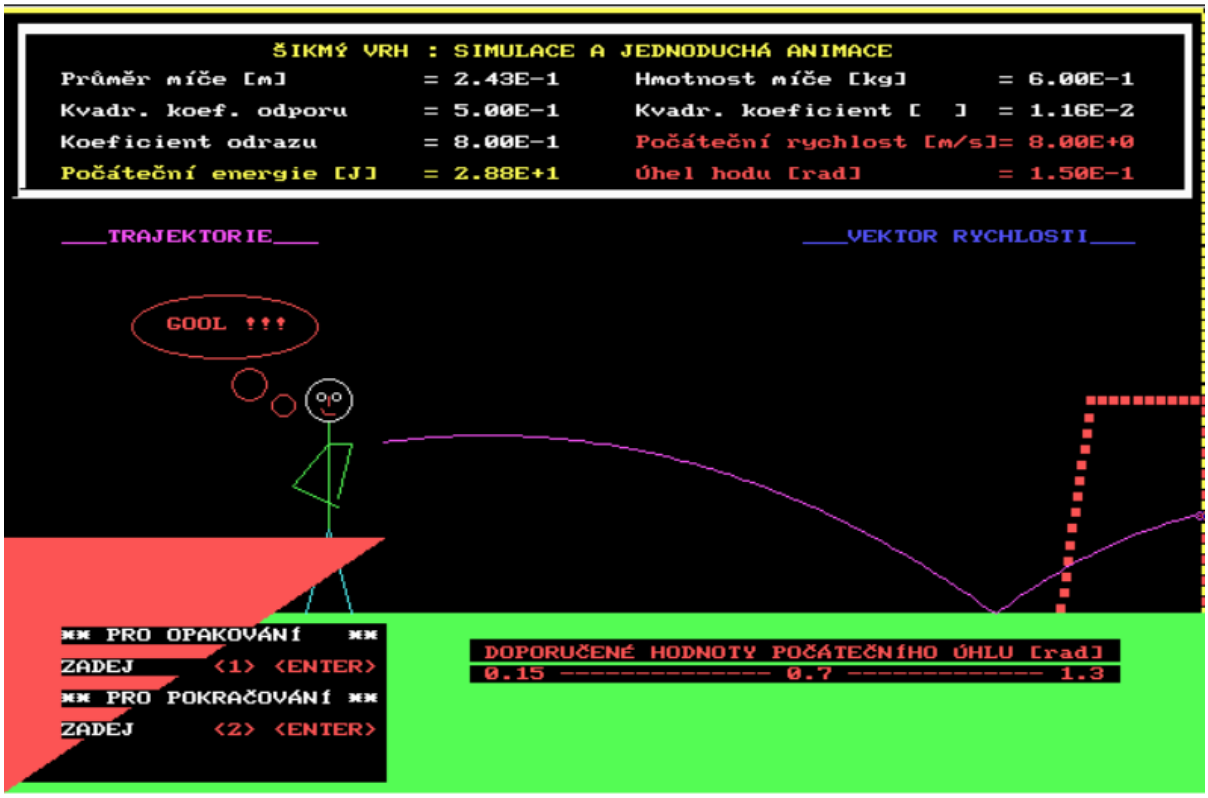
Zadání problémů, které jsou řešeny počítačem, bývá zpravidla slovní. V prvním kroku je důležitá matematizace tohoto problému, tzn. popis reálné situace matematickým zápisem fyzikálního jádra věci. Tím se samozřejmě uživatel dopouští určitého zjednodušení, což většinou znamená, že se neberou v úvahu všechny vlivy související s tímto problémem.

Dalším krokem je sestavení algoritmu pomocí předchozí matematizace. Při tom je podstatné zohlednit, zda získané výsledky budou odpovídat reálné situaci, tedy jestli daný model dostatečně popisuje tuto situaci. Důležité je také zvolit vhodnou metodu k řešení, většina rovnic nemusí být řešitelná jako funkce času.

Posledním neméně důležitým krokem je prezentace získaných hodnot. Nejčastěji se lze setkat s tabulkou či grafem, popřípadě diagramem znázorňujícím časovou závislost vybraných veličin. Ověřovány jsou totiž podstatné hodnoty popisující daný jev, při nedostatečné interpretaci může být daná situace pochopena zcela jinak. (Válek, 2014, str. 35-36)

3 Dosavadní stav využití počítačových modelů ve výuce fyziky

V české republice se využitím počítačových modelů ve fyzice zabývá doc. Dvořák, který spolu se svými kolegy vytvořil výpočetní systém s názvem Famulus. Tento program byl nejvíce využívaným programem na středních školách a využíván byl také na některých vysokých školách.

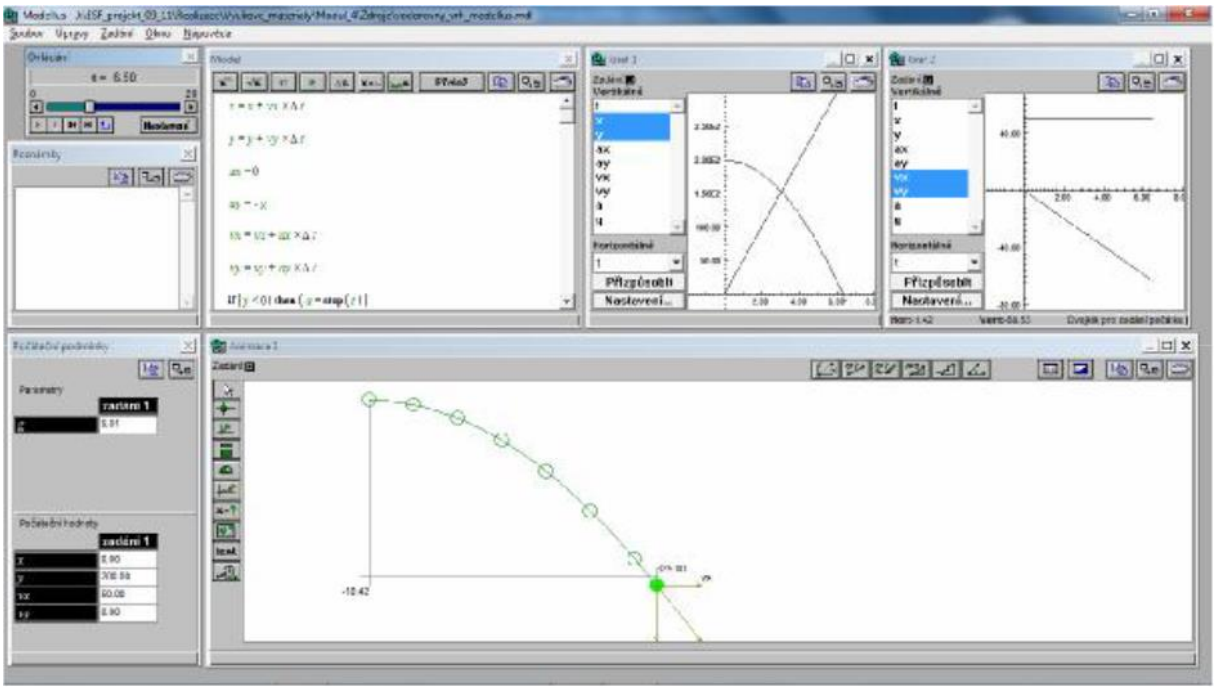


Obrázek 1: Program Famulus. Drska Ladislav

Využívá rozhraní, které je podobné programovacímu jazyku Pascal. Největší výhodou je pak bez pochyby grafický výstup, který má k dispozici širokou nabídku úprav, což umožňuje získání potřebných informací měřených veličin. Program také umožňuje převedení výstupních hodnot do jiných programů. (Dvořák, 1992)

Dalším programem, který se v České republice používá a nahrazuje se jím předchozí program, je Modellus. Využívá opět matematických modelů pro výpočty fyzikálních jevů. Jeho obrovskou výhodou je, že uživatelé nemusí znát ani jeden programovací jazyk, musí se ale orientovat v dané problematice.

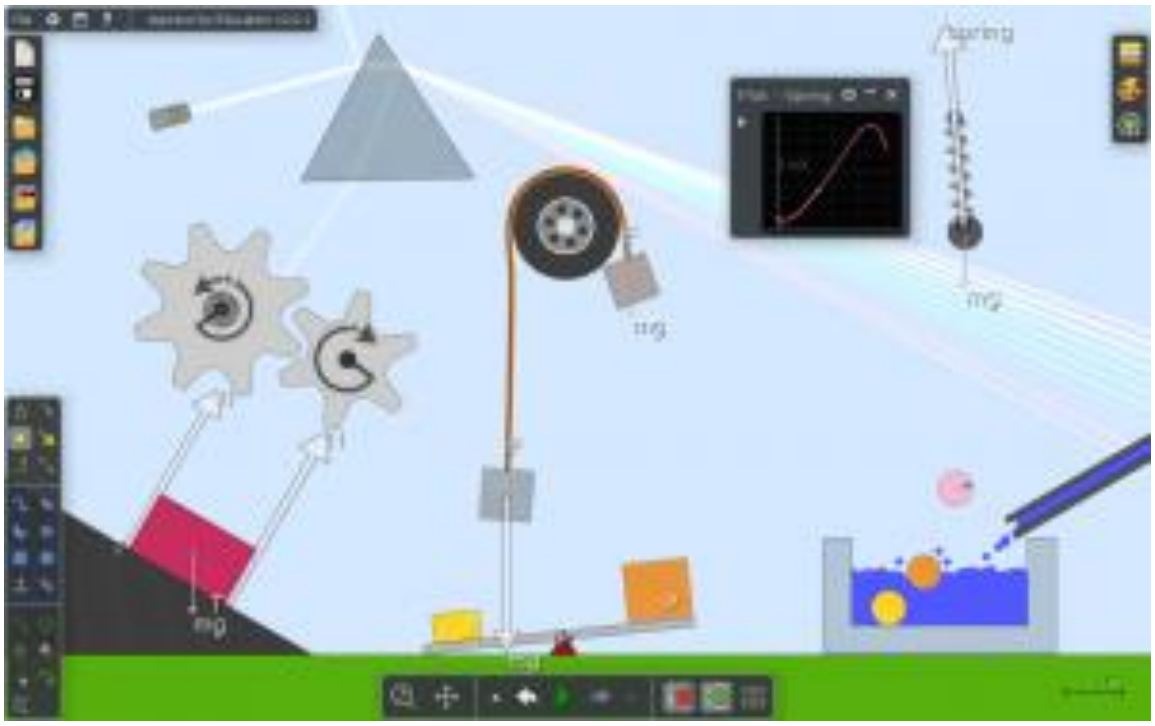
Výstupem pak jsou vypočtené hodnoty zaznamenané do tabulky či grafu, uživatel má možnost vybrat si také z obrázku či videa. Obrazový výstup pak obsahuje zobrazení modelovaných objektů a jejich trajektorií v případě pohybové situace. Může ale obsahovat také vektory veličin, které si sám vybere. Program byl vyroben původně v Portugalsku, přeložen je ale i do českého jazyka, za což vděčíme Martinu Krynickému. (Janeček, 2011)



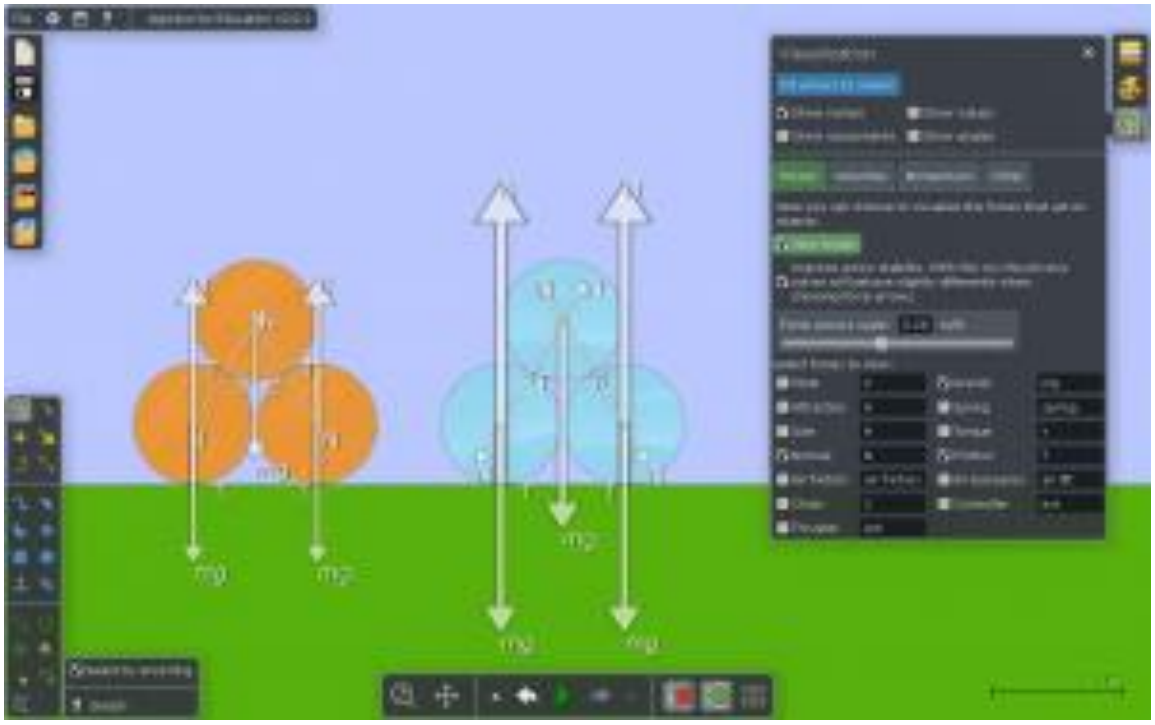
Obrázek 2: Program Modellus. Janeček Petr

Dnes je k modelování využíván program Algodoo, který je také známý pod dřívějším názvem Phun. Hlavním cílem tohoto programu je ukázat fyziku především hravou a také zábavnou formou. Program vznikl v rámci magisterské práce na univerzitě ve Švédsku také za účelem motivace žáků při práci zejména v technických oblastech, konkrétněji pak pro fyzikální vědy. Výhoda tohoto programu je v jeho jednoduchosti, stačí totiž nakreslit zkoumaná tělesa, kterým se posléze přiřadí požadované vlastnosti. Tento program tedy lze využít jak na základní, tak na střední škole a je zaměřen především na mechaniku. Pokud potřebuje uživatel vytvořit cokoli jiného, je potřeba daný problém popsat pomocí mechaniky a poté je Algodoo ideální aplikací pro modelování.

Tato aplikace disponuje také přímou podporou výuky, lze totiž využít připravených vyučovacích hodin. Pedagog je pak může snadno zařadit do hodin. Žáci si mohou programy vytvářet také sami jednoduchým umístěním objektů na scénu a přiřazením konkrétních vlastností, mezi kterými nalezneme například volbu materiálu. Po vybrání daného materiálu počítá Algodoo s konkrétními hodnotami pro hmotnost, pružnost atp. (Černý, 2013, str. 216-223)



Obrázek 3: Program Algodoo - ukázka 1



Obrázek 4: Program Algodoo – ukázka 2

3.1 Unity

3.1.1 Popis programu

V této práci byl k programování zvolených úloh vybrána platforma Unity. Ta se objevila na trhu v roce 2005. Jedná se o herní engine (poskytuje funkce používaných v počítačových hrách), který měl ulehčit vznik a vývoj her i menším skupinám, které se touto problematikou zabývaly. Od té doby tato platforma prošla několika změnami a začala se využívat zejména pro vývojáře her. (Axon, 2016)

Unity již několik let využívá nástroje UnityHub, ve kterém jsou přehledně zaznamenány všechny uživatelem vytvořené projekty. Tento software také obsahuje informace o nainstalovaných verzích (lze mít nainstalováno současně několik verzí). Pod záložkou LEARN pak uživatel může nalézt ukázkové projekty nebo další informace schované pod odkazy.

Jak již bylo zmíněno výše, tento program vznikl především za účelem vytváření her, které obsahují tzv. herní objekty. Pod tímto pojmem si můžeme představit vše, co má pro hru nějaký význam (např. postavy, věci, překážky, …). Tyto objekty se opakovaně pohybují a vykreslují na scéně. Vše tedy lze sledovat, aniž bychom hru spustili, a to přepnutím pomocí záložky.

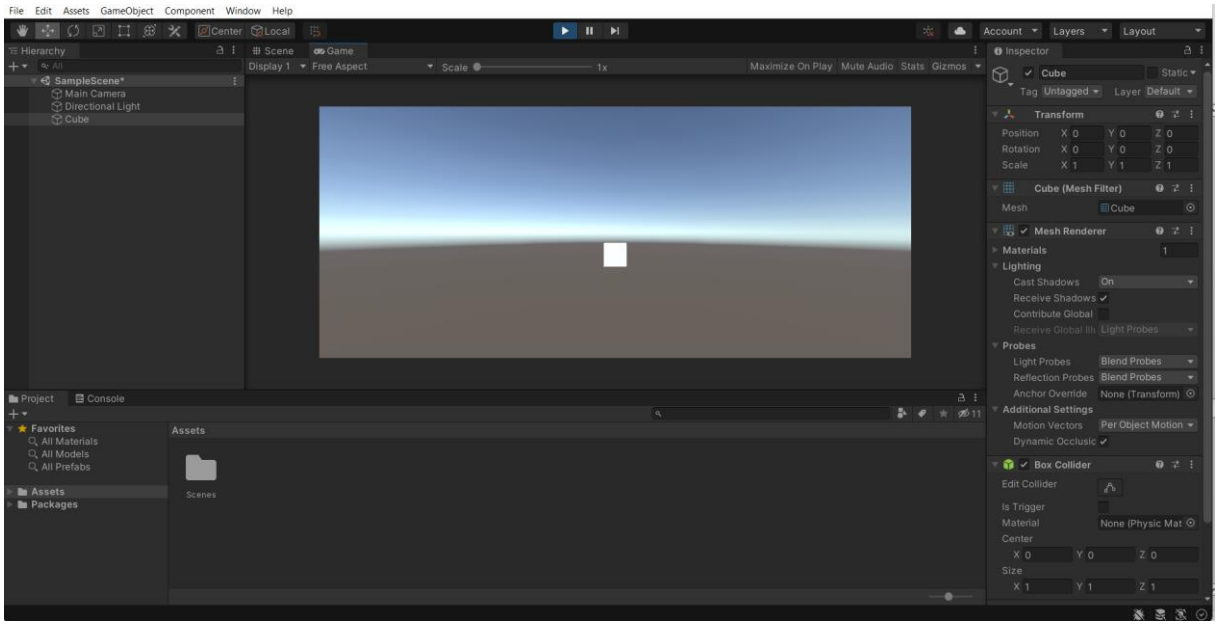
Na této scéně můžeme tedy pozorovat chování herních objektů. Unity umí za uživatele řešit fyziku v podobě detekce kolizí, rychlosti, hybnosti nebo pružnosti herních objektů. U herních objektů lze také změnit materiál, který má požadované fyzikální vlastnosti. Dokáže také upravit tyto parametry v závislosti na požadovaných vlastnostech. Uživatel ale musí vše do programu umět zadat.

Pokud se herní objekty objeví na scéně, nelze s nimi nic provádět. Mluví se o nich jako o „věšácích na vlastnosti“, na které uživatel přiřadí vybrané vlastnosti. Tyto vlastnosti jsou v Unity známy pod pojmem komponenty. Ty může uživatel ovlivnit nastavením jejich parametrů, což lze udělat pomocí panelu Inspektor nebo v pokročilejších fázích pomocí kódu, který bude součástí samotné hry, tzv. skriptu. (Holan, 2020, str. 21-32)

Platforma nabízí dvě varianty, ve kterých lze vytvářet programy. Jedná se o 2D a 3D programování, které se liší rozměry. V Unity 2D lze programovat pouze ve dvou rozměrech. Pokud bychom se chtěli věnovat vytváření aplikací v prostoru, můžeme využít Unity 3D.

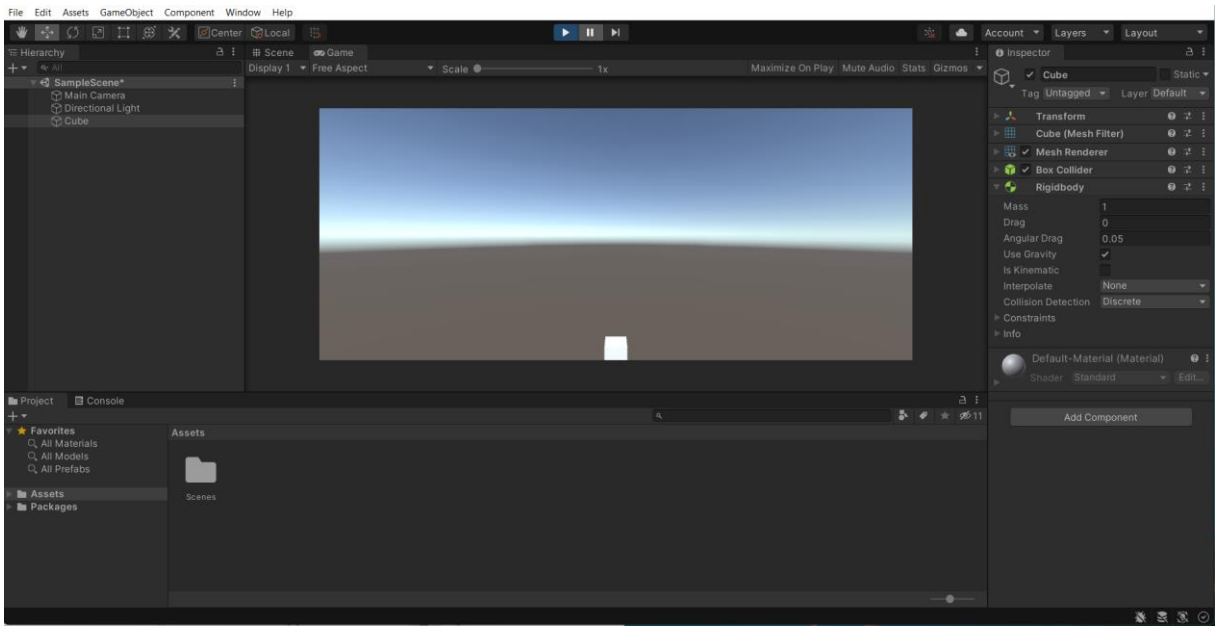
Jednotlivé varianty se téměř neodlišují, každá kategorie má ale své zvláštní označení pro samotné objekty, komponenty, příkazy a další.

Kdybychom tedy chtěli vytvářet vlastní program, spustíme platformu Unity a na scénu umístíme objekt, který reprezentuje například kostka. K dispozici jsou samozřejmě i další varianty, jako například koule, válec nebo těleso ve tvaru tobolky. Tato kostka po spuštění programu neudělá vůbec nic a ani nemůže, protože jsme ji nepřiřadili žádnou vlastnost.



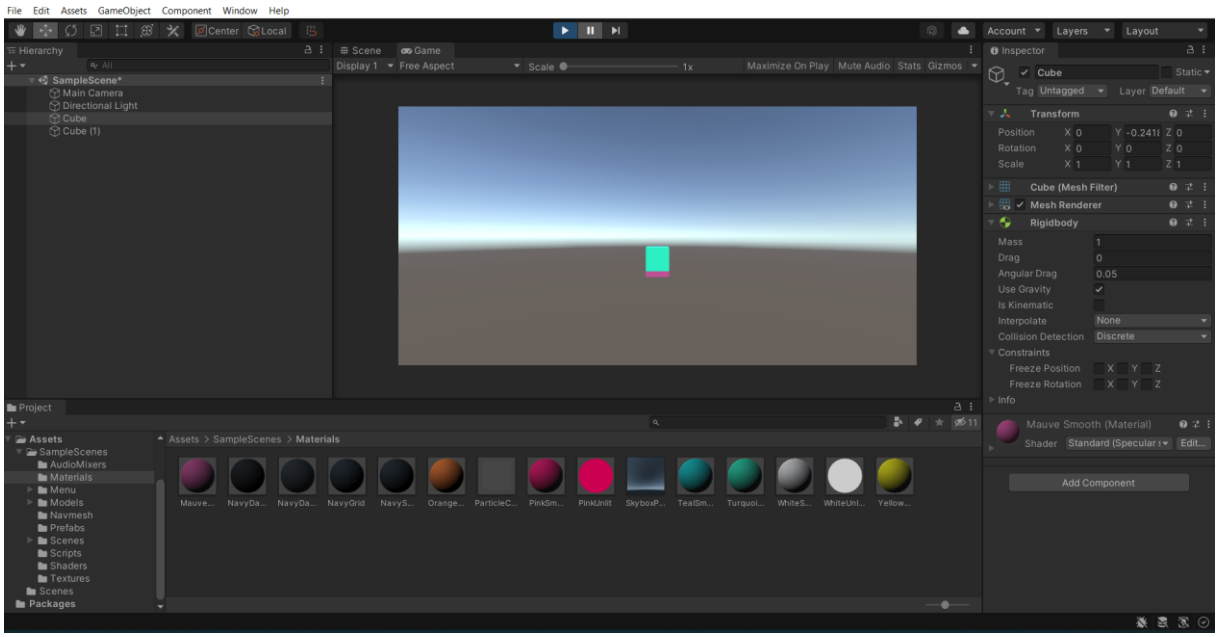
Obrázek 5: Unity - kostka

Pokud požadujeme od kostky, aby po spuštění programu začala něco dělat, je potřeba použít komponentu Rigidbody. Lze ji najít v pravém sloupci kliknutím na Add Component → Physics a následně vybrat zmíněnou vlastnost. Tím kostka nebo jiný vybraný objekt získá hmotnost, která je nastavena na hodnotu 1 nebo například možnost působení gravitace na tento objekt.

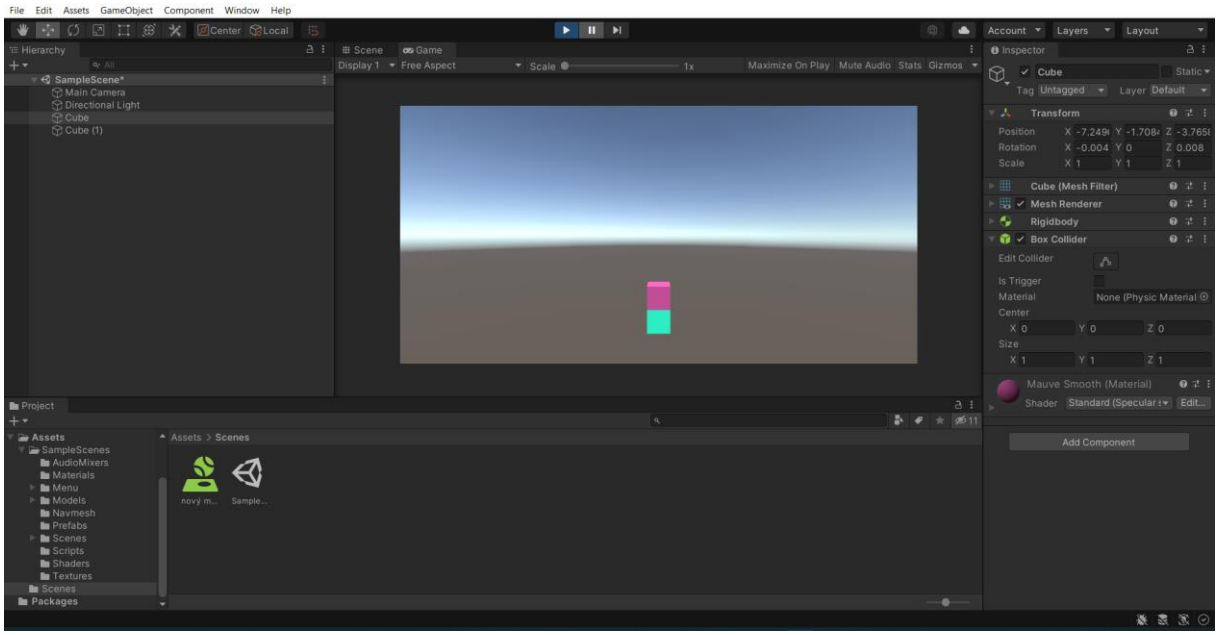


Obrázek 6: Unity – Rigidbody

Po spuštění programu se tedy kostka začne pohybovat působením gravitace směrem dolů. Parametry komponenty Rigidbody lze samozřejmě upravovat a můžeme také zadávat i konkrétní hodnoty. Je ale zřejmé, že od objektu bychom chtěli daleko víc. Budeme chtít například, aby kostka reagovala na jiné objekty (aby docházelo ke kolizím). Musíme využít komponentu Box Collider, ta zajistí to, aby se objekty srazily.

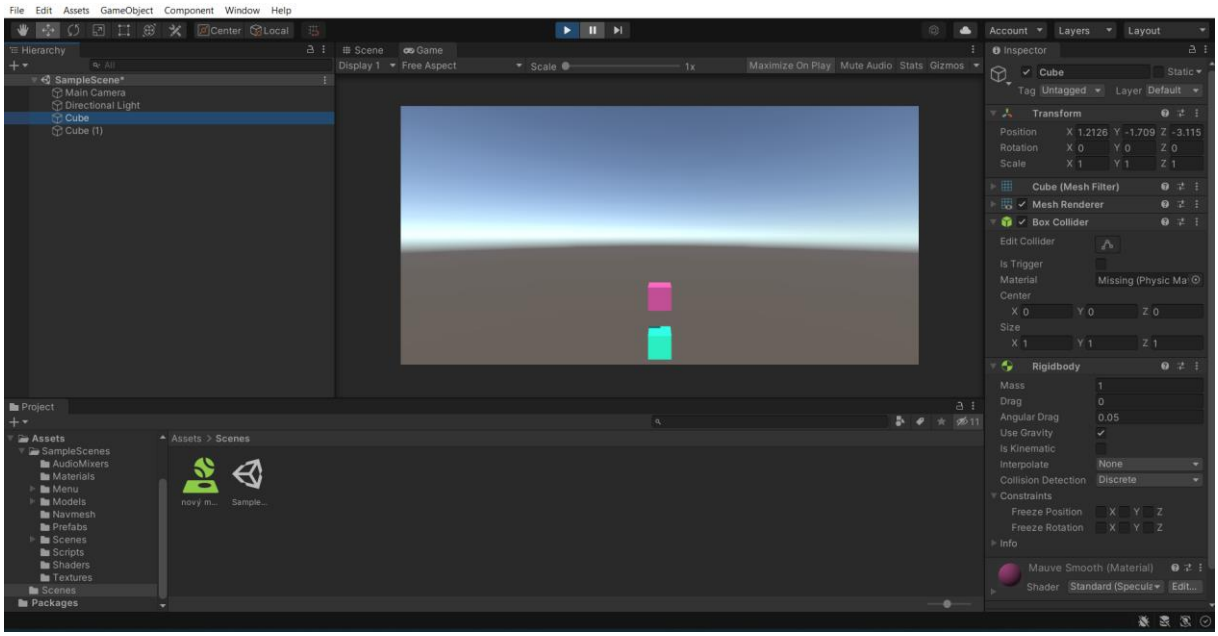


Obrázek 7: Unity - bez Box Collider



Obrázek 8: Unity - s Box Collider

Máme sice zavěšené komponenty, objekty ale nemají přiřazený materiál, což má za následek nepružnou srážku. To lze napravit pomocí tzv. Assets. Ty lze vytvořit úplně nové nebo získat jinými způsobem, například na oficiálních stránkách unity (https://assetstore.unity.com/) stažením a importem. Opět lze upravovat jednotlivé parametry podle potřeby.



Obrázek 9: Unity – Materiál

Výsledkem je pružná srážka a při upravení bychom mohli získat i dokonale pružnou srážku. Podobně bychom mohli přiřazovat další vlastnosti a pověsit je na objekty na scéně.

Tímto způsobem funguje zjednodušeně platforma Unity, i když většina jejích možností nebyla ani zmíněna, natož popsána. (Holan, 2020, str. 21-32)

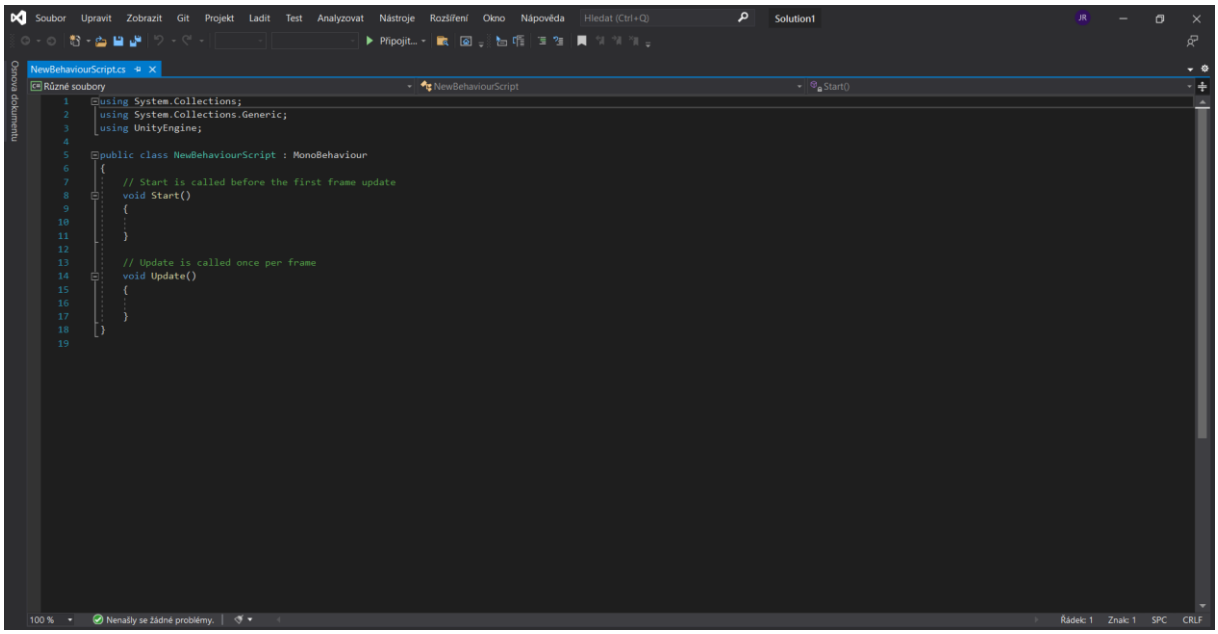
3.1.2 Programovací jazyk

Na hmotné body lze pověsit také kódy, kterými lze docílit například pohybu těchto objektů. V tomto kódu ale potřebujeme objektům na scéně sdělit, co mají dělat. Toho lze dosáhnout pomocí programovacího jazyka, pro Unity se používá jazyk C# (c sharp). Ten vyvinula firma Microsoft. Dnes je nejpoužívanější vývojové prostředí Microsoft Visual Studio, ve kterém lze vytvářet programy právě v tomto jazyce. Tento jazyk vychází především z jazyka C/C++, některé prvky jsou převzaty z jiného programovacího jazyka, kterým je Java.

Mezi základní prvky stěžejního programovacího jazyka platformy Unity se řadí objektová orientovanost. Znamená to, že jazyk C# se zaměřuje především na využití softwarových komponent, které definují objekty na scéně včetně jejich chování.

3.1.3 Struktura programu

Pokud bychom chtěli v Unity vytvořit program, kterým budeme chtít například pohybovat objektem na scéně, nemusí tento objekt na scéně být. Stačí v záložce Assets zvolit možnost Create, kde vybereme C# Script. Tento Script po otevření není prázdný, obsahuje už předepsaný zdrojový kód, u kterého není podstatné, co jednotlivé příkazy znamenají. Vysvětlení nemá pro vytváření programu význam.



Obrázek 10: C# - struktura programu

Důležitější je, že tento programovací jazyk vyžaduje dodržování systému při tvorbě, to znamená psaní příkazů do složených závorek. Další podstatnou věcí je ukončování téměř všech řádků středníkem, výjimkou jsou komentáře, popřípadě jiné konstrukce tohoto programovacího jazyka. Nespornou výhodou při dodržování pravidel je pak obrovská přehlednost ve vytvořeném programu.

Pro dovysvětlení významu jednotlivých řádků se používají komentáře, které mohou být jednořádkové nebo víceřádkové. Pro jednořádkové lze použít dvě lomítka //, za která lze dopsat jakýkoli text. Pokud bychom potřebovali sepsat delší text, je vhodnější použít lomítko a hvězdičku /* pro zahájení psaní a hvězdičku a lomítko */ pro ukončení komentáře. Výhodou komentářů je, že program na ně nereaguje, takže je můžeme vepsat tam, kde budou nejvíce potřeba.

Programátor také velmi často využívá proměnnou. Tento pojem není jednoduché z hlediska programování definovat. Proměnnou bychom mohli popsat přirovnáním k pojmenovanému šuplíku, ve kterém je uložena nějaká konkrétní hodnota. Tyto pojmenované šuplíky pak společně vytvářejí celou kartotéku, pod kterou se skrývá počítačová paměť, ve které si programy uchovávají svá data.

V jazyce C sharp to pak vypadá tak, že kromě názvu pro jednotlivé proměnné je potřeba zapsat také typ hodnot, které tato proměnná bude obsahovat. Je potřeba rozlišovat mezi celým číslem (int), desetinným číslem (float), textem (string) a dalšími. (Bory, 2016, str. 11-32)

4 Modelování vybraných úloh z mechaniky

4.1 Úloha 1 – Slovní úloha o pohybu

Dva hmotné body se pohybují ve směru osy x rovnoměrným přímočarým pohybem tak, že v počátečním čase $t = 0$ jsou od sebe vzdáleny o $s = 60$ m a velikosti jejich rychlostí jsou:

a) $v_{s1} = 5\ \mathrm{m} \cdot \mathrm{s}^{-1}, v_{s2} = 2\ \mathrm{m} \cdot \mathrm{s}^{-1}$;
b) $v_{s1} = 5\ \mathrm{m} \cdot \mathrm{s}^{-1}, v_{s2} = -2\ \mathrm{m} \cdot \mathrm{s}^{-1}$.

Určete místo a čas, ve kterém se v obou případech potkají. (Samek 2014, s. 52)

4.1.1 Řešení

a) V prvním případě se oba hmotné body pohybují stejným směrem. Rychlost jednoho bodu vůči druhému je dána rozdílem, tedy $v_r = 3\ \mathrm{m} \cdot s^{-1}$. To znamená, že každou sekundu se vzdálenost mezi oběma hmotnými body zkrátí o 3 m. Touto úvahou lze snadno určit čas, který bude potřebný k překonání vzdálenosti mezi oběma hmotnými body

$$t_c = \frac{s}{v_r}.$$

Dosazením a výpočtem získáme hodnotu $t_c = 20$ s. Místo, kde se oba hmotné body potkají lze určit dopočtením jedné ze dvou vzdáleností od původní polohy.

První hmotný bod má v čase $t = 0$ hodnotu x-ové souřadnice rovnou 0. Jeho celková vzdálenost je pak rovna

$$l_1 = v_{s1} \cdot t_c.$$

Po dosazení a dopočítání získáme hodnotu $l_1 = 100$ m od počátku.

Stejnou hodnotu od počátku bychom dostali i v případě druhého hmotného bodu. Ten ale urazí jinou vzdálenost než první hmotný bod. Můžeme ji opět dopočítat, tentokrát ale podle vztahu

$$l_2 = s + v_{s2} \cdot t_c.$$

Rozdíl spočívá v tom, že počáteční poloha tohoto hmotného bodu je ve vzdálenosti $60\ \mathrm{m}$ od počátku. Zmíněnou vzdálenost obou bodů tedy započítáme pro určení celkové vzdálenosti. Po dosazení do vztahu uvedeného výše získáme opět hodnotu $l_2 = 100\ \mathrm{m}$, s tím rozdílem, že druhý hmotný bod urazil pouze $40\ \mathrm{m}$.

b) Ve druhém případě se oba hmotné body pohybují opačným směrem, tedy proti sobě. Rychlost jednoho bodu vůči druhému je tedy tentokrát dána součtem obou rychlostí, tedy $v_r = 7\ \mathrm{m} \cdot s^{-1}$. V tomto případě se vzdálenost mezi oběma hmotnými body zkrátí každou sekundu o 7 metrů. Z toho je patrné, že se potkají dříve v místě, které je blíže počátku. Čas zjistíme opět podle vztahu

$$t_c = \frac{s}{v_r}.$$

Po dosazení a zaokrouhlení na dvě desetinná místa získáme hodnotu $t_c = 8{,}57\ \mathrm{s}$. Místo setkání můžeme opět určit dosazením do vztahu pro pohyb prvního nebo druhého hmotného bodu.

Pro první případ tedy platí opět vztah

$$l_1 = v_{s1} \cdot t_c,$$

kde po dosazení a výpočtu získáme hodnotu $l_1 = 42{,}86\ \mathrm{m}$ v případě, že se tento hmotný bod nacházel v počátku souřadnicového systému.

Pro druhý případ pak platí opět stejný vztah jako v situaci a), tzn.

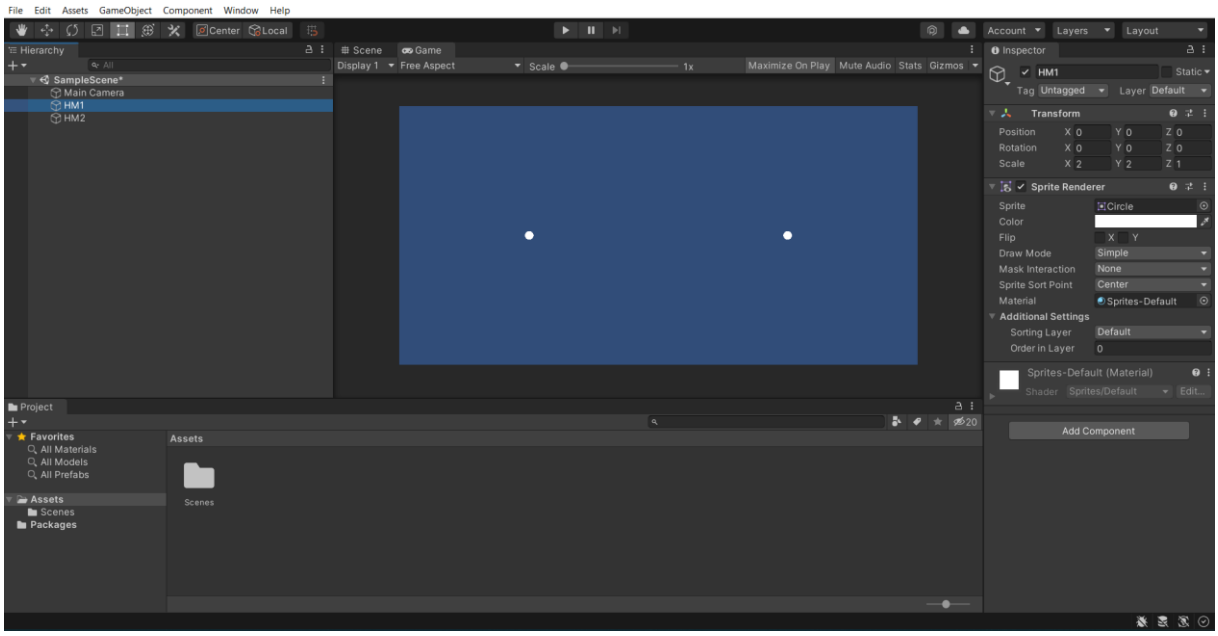
$$l_2 = s + v_{s2} \cdot t_c,$$

kde za rychlost dosazujeme zápornou rychlost, tedy v opačném směru pohybu prvního hmotného bodu. Po dosazení a vypočtení získáme opět hodnotu $l_2 = 42{,}86\ \mathrm{m}$, což znamená, že druhý hmotný bod urazil vzdálenost $17{,}14\ \mathrm{m}$ směrem k počátku soustavy.

4.1.2 Modelace v Unity

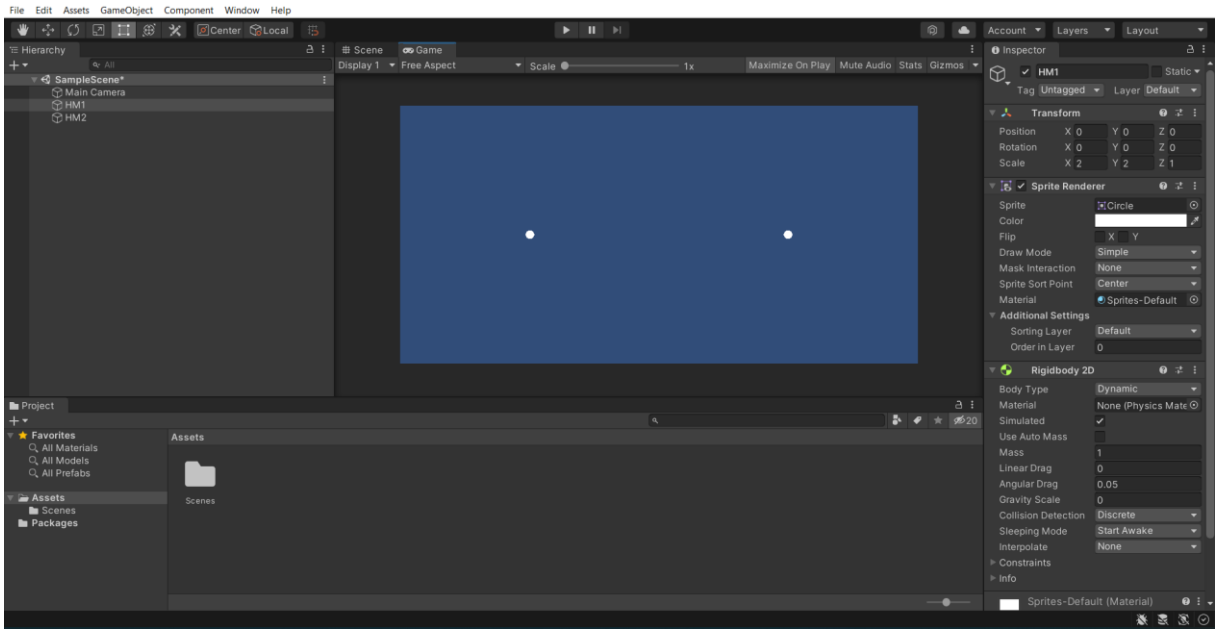
Úloha byla namodelována v platformě Unity. Modelace probíhala vzhledem k zadání v Unity 2D. Základem je umístění dvou objektů, kterými budou kruhy. Tyto kruhy pro přehlednost pojmenujeme koule1 a koule2, které reprezentují hmotný bod 1, resp. 2. Nyní je umístíme vzhledem k zadání šedesát jednotek od sebe. Prvnímu hmotnému bodu byla tedy hodnota x-ové souřadnice nastavena na -50, jeho souřadnice bychom mohli zvolit i jiné a podle nich upravit hodnoty druhého hmotného bodu. Abychom dodrželi zadání úlohy 1, x-ová souřadnice druhého hmotného bodu byla nastavena na hodnotu 10.

Pro zobrazení scény je ještě velmi důležité upravit její velikost. Právě proto jsme nastavovali výchozí souřadnice obou hmotných bodů na výše zmíněné hodnoty. Šestinásobným zvětšením scény získáme na hodnotu 30 získáme scénu o rozměrech 60 jednotek na 60 jednotek (Main Camera: Size: $5 \rightarrow 30$). Tím máme zajištěný dostatečný prostor pro sledování celé situace. Velikosti hmotných bodů byly později upraveny na trojnásobek.



Obrázek 11: Úloha 1 – Scéna 2D

Na hmotné body zatím nic nepůsobí, ale ze zadání víme, že se tyto hmotné body mají pohybovat různými rychlostmi a různým směrem. Využijeme tedy zmíněné komponenty RigidBody, tentokrát ale nezaškrtneme možnost gravitace (v Unity 2D vypneme gravitaci změnou parametru Gravity Scale z hodnoty 1 na 0).



Obrázek 12: Úloha 1 - Rigidbody 2D

Na scéně tedy máme umístěné body, kterými budeme chtít pohybovat a na ně zavěšené komponenty Rigidbody 2D, které tento pohyb umožní. Přichází tedy část, která bude stěžejní pro vytvoření animace. Tou je vytváření příkazu v programovacím jazyce C#.

V záložce Assets zvolíme možnost Create, ze které vybereme C# Script, což bude programovací nástroj pro zmíněný pohyb. Po naprogramování výše uvedené změny souřadnic a následném spuštění programu zjistíme, že se objekty na scéně nehýbou. Objekty na scéně jsou „věšáky“ a naším úkolem je na ně pověsit zmíněný C# Script. Nyní bychom měli tento pohyb naprogramovat.

Nejprve budeme potřebovat „zavolat“ jednotlivé objekty a komponenty ze scény. Tento krok je velmi důležitý, protože musíme vědět, k čemu se mají další kroky vázat:

```
koule1 = GameObject.Find("koule1");
component1 = koule1.GetComponent<Rigidbody2D>();
koule2 = GameObject.Find("koule2");
component2 = koule2.GetComponent<Rigidbody2D>();.
```

Toto zavolání provádíme příkazem GameObject.Find a v závorce uvedeme objekt, který chceme zavolat (koule1 nebo koule2). Pro přehlednost si jednotlivé příkazy pojmenujeme.

Následně budeme využívat výše zmíněné komponenty Rigidbody 2D, které získáme příkazem GetComponent<Rigidbody2D>(). Nesmíme zapomenout ukončovat každý řádek středníkem.

Nyní bude naším úkolem rozpohybovat tyto hmotné body podle zadání. Modelace bude probíhat pro oba případy. K pohybu bude potřeba vytvořit příkaz zajišťující posunutí souřadnic obou hmotných bodů za čas. Budeme požadovat, aby se toto posunutí opakovalo, to zajistíme neustálým přičítáním těchto „kousků" k předchozím hodnotám.

Prvním krokem této části bude vytvoření proměnných hodnot, což zajistíme vytvořením nových vektorů příkazem Vector2. Na počátku nastavíme veškeré hodnoty na 0. Budeme totiž chtít, aby se hmotné body začaly pohybovat až budeme pohyb požadovat (po stisknutí tlačítka).

```
Vector2 rychlost1 = new Vector2(0, 0);
Vector2 rychlost2 = new Vector2(0, 0);
```

Následovat bude samotná změna souřadnic. Pro rychlost bude platit, že začneme přičítat k původním souřadnicím rychlost, kterou později zadáme, vynásobenou časovým úsekem, který zvolíme v nastavení Unity.

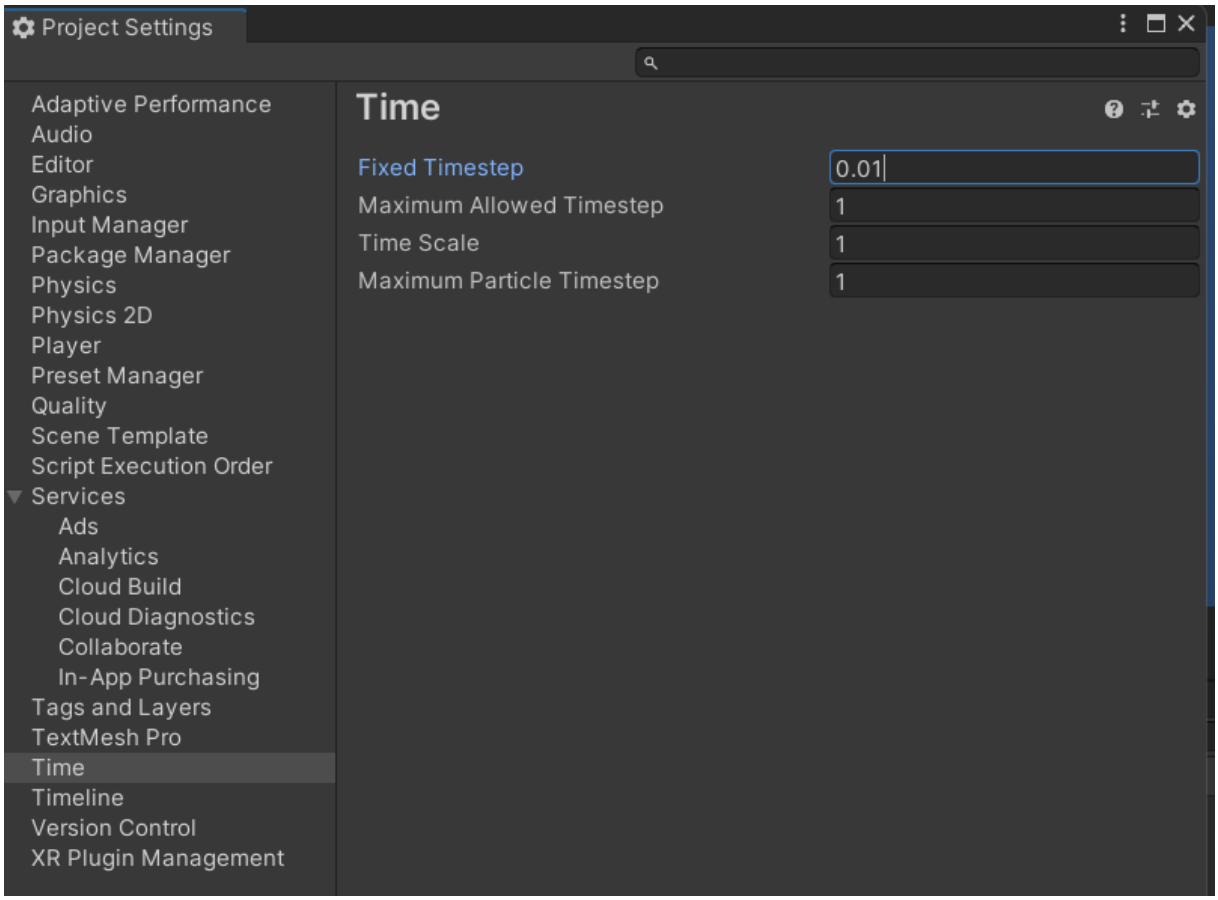
Tato část zdrojového kódu vypadá pak následovně:

```
component1.transform.position = new Vector2
(component1.transform.position.x + rychlost1.x
*Time.fixedDeltaTime, component1.transform.position.y + rychlost1.y
*Time.fixedDeltaTime);
component2.transform.position = new Vector2
(component2.transform.position.x + rychlost2.x
*Time.fixedDeltaTime, 0);.
```

Component1.transform.position.x znamená změnu x-ové souřadnice, ke které přičteme zmíněný součin rychlosti ve směru osy x s časovým úsekem. Obdobně bychom mohli postupovat ve směru y, který ale pro nás nebude relevantní. Proto je v druhém případě y-ová souřadnice nastavena na hodnotu 0.

Time.fixedDeltaTime je v tomto případě časový interval mezi „pevnými“ snímky, který je udáván v sekundách. Tento časový údaj není závislý na snímkovací frekvenci vytvořené animace. Jeho hodnota je konstantní a v nastavení má výchozí hodnotu 0.02, což znamená, že program je aktualizován 50krát za sekundu.

Nastavení časového úseku lze provést kliknutím na Edit → Project Settings. Tím se otevře tabulka, ve které můžeme nastavovat spoustu věcí včetně času. Vybereme tedy možnost Time a zobrazí se nám celkem 4 položky, které můžeme upravovat. Nás bude zajímat položka Fixed Timestep, která udává velikost jednoho časové skoku. Pro zajištěním plynulého posunu hmotných bodů byla nastavena hodnota na 0.01 $(100 \text{ krát za sekundu})$. Pokud bychom nastavili hodnotu na 0.1 nebo dokonce 1, viděli bychom jednotlivé skoky.



Obrázek 13: Nastavení časového skoku

Tento časový úsek souvisí s příkazem FixedUpdate, který se používá zejména pro fyziku. Jedná se o příkaz, kdy je volána aktualizace snímku. Jeho výhodou je, že není závislý na snímkové frekvenci.

Chybí nám ještě nastavení rychlostí pro oba hmotné body. Vhodné řešení se nabízí zadáváním a případnou změnou v průběhu animace. Přidáme tedy na scénu InputField (vstupní pole), který lze nalézt pod záložkou Component → UI. Pojmenujeme tuto komponentu a přesuneme se opět do C# Scriptu.

Obrázek 14: InputField pro zadávání rychlostí

Je zřejmé, že do komponenty InputField budeme zadávat číselnou hodnotu. V C#Scriptu proto bude nutné text převést na číslo, což zajistíme příkazem float.Parse. Musíme ale nejprve vědět, co chceme převést, takže nesmíme zapomenou opět „zavolat“ objekt a vybrat to, co pro nás bude podstatné. Govelocity je pouze pojmenování pro tento herní objekt.

```
GameObject govelocity1 = GameObject.Find("rych1");
        InputField text = govelocity1.GetComponent<InputField>();
        if (text != null)
        {
            string stext = text.text;
            rychlost1.x= float.Parse(stext);
        }
        GameObject govelocity2 = GameObject.Find("rych2");
        text = govelocity2.GetComponent<InputField>();
        if (text != null)
        {
            string stext = text.text;
            rychlost2.x = float.Parse(stext);
```

```
}
```

V tento moment jsme schopni rozpohybovat oba hmotné body zadanými rychlostmi. Chtěli bychom ale také sledovat čas a dráhy obou hmotných bodů v průběhu animace. Přidáme tedy ještě 3 texty, které budou zobrazovat uběhnutý čas a dráhy obou hmotných bodů.

Obrázek 15: Zobrazovací texty

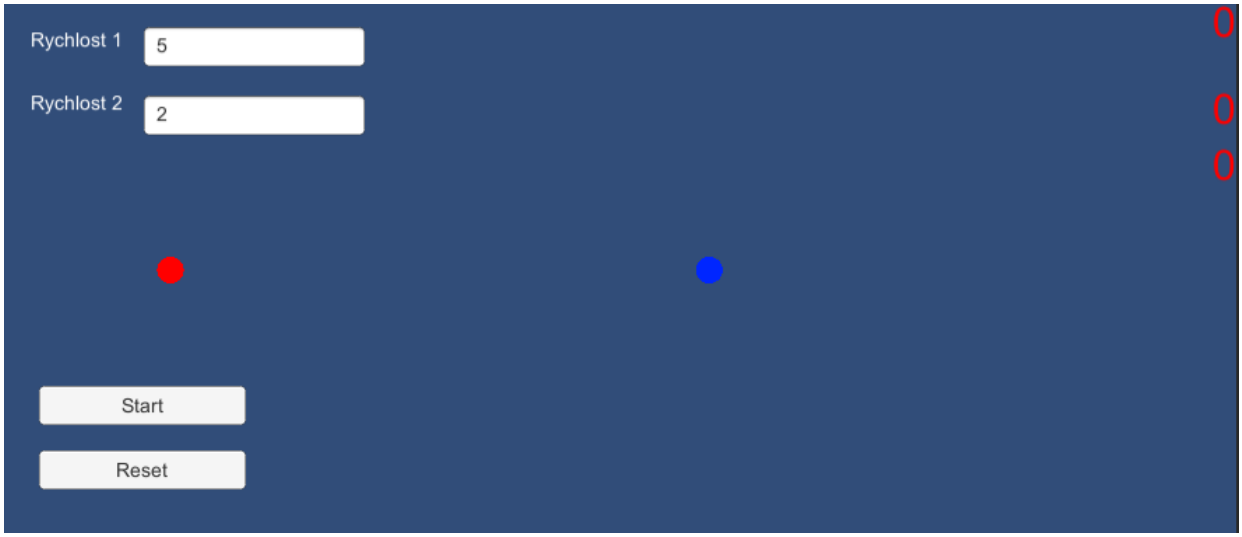
```
if (beforeCrash)
{
    txt.text = startTime != DateTime.MinValue ?
sec.ToString() : "0";
    txt = GameObject.Find("draha1").GetComponent<Text>();
    txt.text = (rychlost1.x * sec).ToString();
    txt = GameObject.Find("draha2").GetComponent<Text>();
    txt.text = (rychlost2.x * sec).ToString();
}
```

Budeme chtít počítat čas a dráhy až do okamžiku, kdy se oba hmotné body potkají. Využili jsme příkazu bool, který může nabývat pouze hodnoty false nebo true. Po spuštění animace bude nastavena hodnota false a v případě, že hodnota souřadnice x prvního hmotného bodu bude větší než hodnota souřadnice x druhého hmotného bodu, tyto texty přestanou počítat. Zobrazí pak hodnotu, při které došlo „ke srážce“.

Čas počítá hodnotu od okamžiku zahájení do srážky, draha1 a draha2 pak počítá celkovou uraženou dráhu hmotným bodem 1, resp. hmotným bodem 2. Dráha jedna se počítá jako rychlost1 ve směru x vynásobená aktuální časovou hodnotou, obdobně je to s dráhou dva.

Pro zobrazení potřebujeme text, takže tyto číselné hodnoty zobrazíme pomocí ToString() jako text.

Poslední věcí je, že chceme program spustit pomocí tlačítka. Přidáme jej na scénu opět přes Component → UI, ale tentokrát zvolíme možnost Button (tlačítko). Přidáme ještě jedno tlačítko na resetování animace. Obě tlačítka pojmenujeme.



Obrázek 16: Tlačítka Start a Reset

To, co mají tlačítka dělat bude nutné popsat do scriptu.

```
public void ButtonPressed()
    {
        UnityEngine.Debug.Log("Pressed");
        startTime = DateTime.Now;
```

Stisknutím tlačítka Start se spustí časomíra. Je zřejmé, že po zmáčknutí budeme chtít také rozpohybovat oba hmotné body. Proto výše zmíněné příkazy pro govelocity1 a govelocity2 přesuneme do této oblasti.

Tlačítko Reset bude mít pak za úkol vrátit vše do původního stavu. Oba hmotné body do výchozí polohy, vynulovat počáteční rychlosti i hodnoty textů. To zajistíme už velmi snadno, pro rychlosti zavedeme nový vektor2, jehož souřadnice budou nulové. Čas a dráhy vynulujeme tím, že nastavíme časovou hodnotu na minimální. Zbývá nám posunout hmotné body do původní polohy, takže je transformujeme pomocí dvourozměrného vektoru do příslušných pozic. Souřadnice přitom známe.

```
public void ResetButton()
    {
        UnityEngine.Debug.Log("Pressed reset");
        rychlost1 = new Vector2(0, 0);
        rychlost2 = new Vector2(0, 0);
        beforeCrash = true;
        koule1.transform.position = new Vector2(-50, 0);
        koule2.transform.position = new Vector2(10, 0);
        startTime = DateTime.MinValue;
    }
```

Nesmíme zapomenout na všechny objekty pověsit C#Script. Pokud to neuděláme, nic z toho, co jsme tu dosud popisovali, nebude fungovat. Celý C# Script je uveden v příloze jedna.

4.2 Úloha 3 – Dokonale pružná srážka vozíků

Na vzduchové dráze se srazí vozík dokonale pružně s druhým vozíkem, který byl do srážky v klidu. Po srážce se oba vozíky pohybují stejně velkými rychlostmi opačným směrem. Určete poměr hmotností obou vozíků. (Řešené úlohy, 2006)

4.2.1 Řešení úlohy

Za předpokladu, že se vozíky pohybují po přímce budeme postupovat obdobně jako v příkladu 2, s tím rozdílem, že budeme muset pro výpočet použít také zákon zachování mechanické energie. Opět tedy platí ze zákona zachování hybnosti, že je součet hybností v izolované soustavě konstantní, navíc bude konstantní také součet kinetické a potenciální energie (ze zákona mechanické energie).

Označme opět pro přehlednost fyzikální veličiny důležité pro výpočet úlohy:

m_1 … jako hmotnost prvního vozíku,

m_2 … jako hmotnost druhého vozíku,

$\overrightarrow{v_1}$ … jako rychlost prvního vozíku před srážkou,

$\overrightarrow{v_2}$ … jako rychlost druhého vozíku před srážkou,

$\overrightarrow{v_1}'$ … jako rychlost prvního vozíku po srážce,

$\overrightarrow{v_2}'$ … jako rychlost prvního vozíku po srážce,

$\overrightarrow{p_1}$ … jako hybnost prvního vozíku před srážkou,

$\overrightarrow{p_2}$ … jako hybnost druhého vozíku před srážkou,

$\overrightarrow{p_1}'$ … jako hybnost prvního vozíku po srážce,

$\overrightarrow{p_2}'$ … jako hybnost druhého vozíku po srážce,

E_{k1} … jako kinetickou energii prvního vozíku před srážkou,

E_{k2} … jako kinetickou energii druhého vozíku před srážkou,

E'_{k1} … jako kinetická energii prvního vozíku po srážce,

E'_{k2} … jako kinetickou energii druhého vozíku po srážce.

Pro řešení bude vhodné vyjádřit rychlosti před srážkou a po srážce vozíků. Před srážkou bude mít druhý vozík nulovou rychlost, tedy:

$$\overrightarrow{v_2} = 0\ \mathrm{m} \cdot \mathrm{s}^{-1},$$

po srážce je pak rychlost vozíků stejná, což můžeme zapsat takto:

$$v_1{}' = v_2{}', \overrightarrow{v_1}' = -\overrightarrow{v_2}'.$$

Pro dokonale pružnou srážkou platí zmíněný zákon hybnosti, který zapíšeme ve tvaru:

$$\overrightarrow{p_1} + \overrightarrow{p_2} = \overrightarrow{p_1}' + \overrightarrow{p_2}'.$$

Dále použijeme vzorec pro hybnost ($p = mv$), který dosadíme do této rovnice:

$$m_1\overrightarrow{v_1} + m_2\overrightarrow{v_2} = m_1\overrightarrow{v_1}' + m_2\overrightarrow{v_2}'.$$

Zavedeme souřadný systém, pomocí kterého přepíšeme rovnici skalárně. Nejvhodnější bude orientovat osu x ve směru jízdy prvního vozíku před srážkou, tedy druhého vozíku po srážce. Využijeme také toho, že je rychlost druhého vozíku před srážkou nulová a že jsou rychlosti po srážce stejně velké, to znamená:

$$m_1 v_1 = -m_1 v'_1 + m_2 v'_1.$$

Vyjádřit poměr hmotností z tohoto vztahu lze, jenže neznáme rychlosti. Budeme tedy potřebovat aplikovat zákon zachování mechanické energie, který má tvar:

$$E_{k1} + E_{k2} = E'_{k1} + E'_{k2}.$$

Opět použijeme vzorec, tentokráte pro kinetickou energii ($E_k = \frac{1}{2}mv^2$) a dosadíme, využijeme opět toho, že jsou rychlosti po srážce stejně velké a také toho, že rychlost druhého vozíku je před srážkou nulová:

$$\frac{1}{2}m_1v_1^2 + 0 = \frac{1}{2}m_1v_1'^2 + \frac{1}{2}m_2v_1'^2.$$

Upravíme vynásobením dvěma:

$$m_1v_1^2 = m_1v_1'^2 + m_2v_1'^2.$$

Nyní máme k dispozici soustavu rovnic, ze kterých se budeme snažit získat řešení:

$$\text{ZZH: } m_1v_1 = -m_1v_1' + m_2v_1',$$
$$\text{ZZME: } m_1v_1^2 = m_1v_1'^2 + m_2v_1'^2.$$

Z obou rovnic vytkneme a první rovnici umocníme:

$$\text{ZZH: } m_1^2v_1^2 = v_1'^2(m_2 - m_1)^2,$$
$$\text{ZZME: } m_1v_1^2 = v_1'^2(m_1 + m_2).$$

Vyjádříme dále z obou rovnic rychlost po srážce $v_1'^2$ a rovnice srovnáme:

$$\frac{m_1^2v_1^2}{(m_2 - m_1)^2} = \frac{m_1v_1^2}{(m_1 + m_2)}.$$

Obě strany rovnice vydělíme nenulovým výrazem $m_1v_1^2$ a upravíme do tvaru:

$$m_1(m_1 + m_2) = (m_2 - m_1)^2.$$

Posledním krokem je úprava této rovnice:

$$m_1^2 + m_1m_2 = m_2^2 - 2m_1m_2 + m_1^2,$$

$$m_2^2 - 3m_1m_2 = 0,$$

$$m_2(m_2 - 3m_1) = 0.$$

Z této rovnice zjistíme poměr mezi hmotnostmi vozíků. Víme totiž, že hmotnost druhého vozíku je nenulová, zbývá dořešit nulovost závorky:

$$m_2 = 3m_1,$$

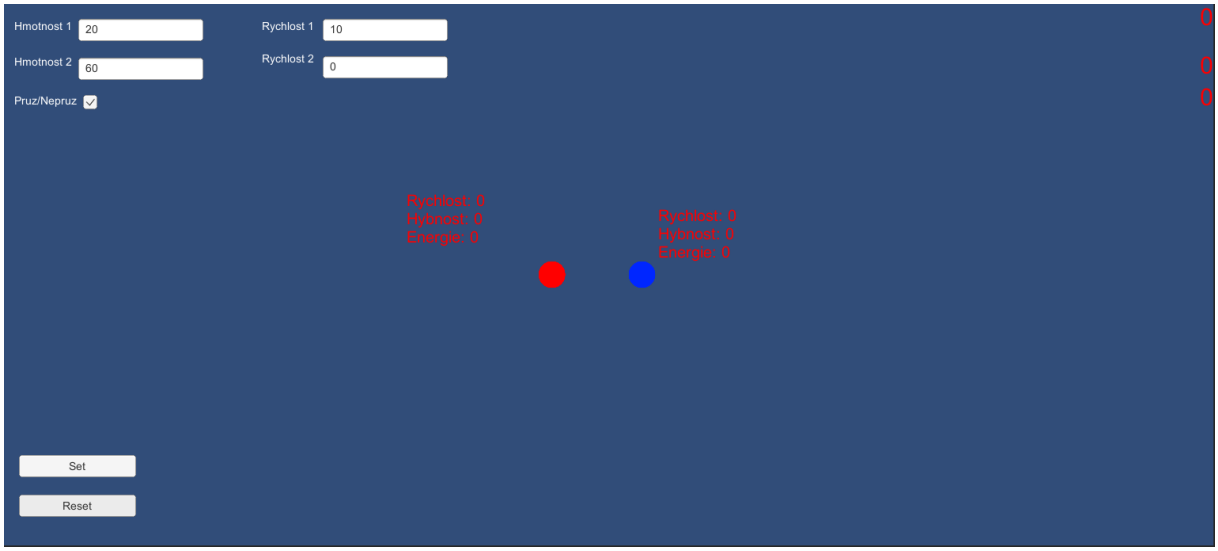
$$\frac{m_2}{m_1} = 3.$$

Poměr mezi hmotnosti vozíků je roven třem.

4.2.2 Modelace v Unity

Modelace této úlohy bude vycházet z úlohy 1, využijeme toho, co jsme doposud vytvořili. Postupně budeme upravovat scénu a doplňovat do scriptu jednotlivé příkazy pro přidané objekty. Tato část bude opět doplněna o ukázky příkazů použité pro tuto úlohu.

Vycházejme tedy ze scény, kterou máme k dispozici z příkladu jedna. Už víme, jak se na scénu umisťují objekty a komponenty, jak se pojmenovávají a jak se „volají“ v scriptu. Vše je podrobně rozepsáno v podkapitole 5.1.2. Na tuto scénu umístíme další InputField, do kterých budeme tentokrát zadávat hmotnosti a doplníme je pro přehlednost popisky. Dále přidáme jednotlivým koulím zvláštní popisky, u kterých budeme chtít zobrazovat rychlost, hybnost a energii, kterou budou mít před srážkou a po srážce. Poslední úpravou bude předělání textů, které počítaly dráhy obou koulí. Ty už nebudeme potřebovat, ale můžeme místo nich počítat celkovou hybnost a celkovou hmotnost. Scéna na obrázku je již upravená.



Obrázek 17: Upravená scéna pro úlohu2

V této scéně se navíc objevuje tlačítko pro přepínaní pružné a nepružné srážky. Tohoto tlačítka si zatím nebudeme všímat, bude potřeba až k úloze 3. Tím vytvoříme jednu aplikaci,

ve které budeme moci přepínat mezi pružnou a nepružnou srážkou v závislosti na zadání úlohy.

Stěžejními pro popis úlohy 2 budou samozřejmě příkazy ve scriptu. Opět nesmíme zapomenout „pověsit“ na nově přidané objekty a komponenty tento script, aby vše fungovalo. Nyní bude následovat popis příkazů, které jsou důležité pro chod úlohy.

```
GameObject gomass1 = GameObject.Find("mass1");
InputField text = gomass1.GetComponent<InputField>();
if (text != null)
{
 string stext = text.text;
 hmotnost1 = float.Parse(stext);
}

GameObject gomass2 = GameObject.Find("mass2");
text = gomass2.GetComponent<InputField>();
if (text != null)
{
 string stext = text.text;
 hmotnost2 = float.Parse(stext);
}
```

Pro popis energie a hybnosti budeme potřebovat znát hmotnosti obou hmotných bodů. Proto jsme na scénu umístili další dvě komponenty s názvem InputField. Do těchto polí budeme zadávat hmotnosti obou koulí. Hodnotu potřebujeme převést na číslo, k čemuž slouží příkaz float.Parse. Pojmenujeme si nově získané hodnoty tak, abychom je mohli využívat v další části příkazu.

Znovu bude nutné rozlišit dvě situace. Jedna bude před srážkou koulí, druhá po srážce. K tomu, aby ke srážce došlo, bude nutné srážku vytvořit. Máme dvě koule o poloměru 3 jednotky, u kterých transformujeme souřadnice ve směru x. Srážku tedy můžeme zajistit tak, že nastavíme podmínku na jejich souřadnice.

```
if (beforeCrash && (koule1.transform.position.x + 3 > koule2.transform.position.x)) //srazka

beforeCrash = false;

float v1 = rychlost1.x;
float v2 = rychlost2.x;
```

```
float v1new, v2new;
```

Pokud tedy nastane situace, ve které hodnota souřadnice x u koule jedna bude o 3 jednotky menší než hodnota souřadnice x u koule dva, systém tuto situaci vyhodnotí jako srážku a nastaví hodnotu beforeCrash na false. Pro popis celé situace označíme pro přehlednost původní rychlosti jako v1 a v2 a rychlosti po srážce jako v1new a v2new. Všechny veličiny jsou zavedené jako float, mohou tedy nabývat desetinných čísel. Původní rychlosti obou koulí budou využívat již popsaných rychlostí ve směru x.

Dále je tedy potřeba do příkazu uvést, jak mají nové rychlosti vypadat. Vyjdeme z toho, že dochází k pružné srážce, ve které platí zákon zachování hybnosti a zákon zachování energie. Z těchto rovnic získáme pomocí původních rychlostí a jednotlivých hmotností rychlosti obou koulí po srážce. Tyto rychlosti zapíšeme do scriptu. Nesmíme zapomenout, že se rychlosti jednotlivých koulí změní, což můžeme zapsat novým vektorem, ve kterém bude první souřadnice odpovídat novým rychlostem.

```
v1new = (hmotnost1 * v1 + hmotnost2 * (2 * v2 - v1)) / (hmotnost1 + hmotnost2);
v2new = (hmotnost2 * v2 + hmotnost1 * (2 * v1 - v2)) / (hmotnost1 + hmotnost2);

rychlost1 = new Vector2(v1new, 0);
rychlost2 = new Vector2(v2new, 0);
```

Tím máme zajištěný pohyb koulí před a po srážce. Pro přehlednost budeme chtít zobrazovat jednotlivé rychlosti, hybnosti a energie. Proto jsme přidali popisky k jednotlivým objektům, které vypadají následovně:

```
GameObject.Find("popisRychlost1").GetComponent<Text>().text = "Rychlost: " + rychlost1.x.ToString() + "\nHybnost: " + (hmotnost1 * rychlost1.x).ToString() + "\nEnergie: " + (0.5 * hmotnost1 * rychlost1.x * rychlost1.x).ToString();

GameObject.Find("popisRychlost2").GetComponent<Text>().text = "Rychlost: " + rychlost2.x.ToString() + "\nHybnost: " + (hmotnost2 * rychlost2.x).ToString() + "\nEnergie: " + (0.5 * hmotnost2 * rychlost2.x * rychlost2.x).ToString();
```

Nejprve opět „zavoláme“ popisek, do kterého postupně uvedeme rychlost, hybnost a energii. Rychlost připojíme příkazem rychlost1.x.ToString(). Pro psaní hybnosti pak platí,

že ji získáme tak, že aktuální hmotnost1 vynásobíme rychlostí jedna ve směru x a vše převedeme na text. Kinetická energie pak má tvar $\frac{1}{2}\mathrm{m}_1{\mathrm{v}_1}^2$ znovu s příponou ToString(). To samé platí pro druhý popisek ovšem s indexy 2. Přidali jsme příkazy, které rozpohybovaly popisky společně s koulemi. Ty budou součástí zdrojového kódu k úloze 2 a 3 v příloze.

Popisky pro dráhy obou koulí v tomto případě postrádají smysl, proto je upravíme. Tyto popisky budou nyní počítat celkovou hybnost a celkovou energii. Celkovou hybnost i energii spočteme podle vztahů uvedených v příkazu. Vše opět převedeme na text.

```
txt = GameObject.Find("kin").GetComponent<Text>();
            txt.text = (0.5f * hmotnost1 * (rychlost1.x *
rychlost1.x) + 0.5f * hmotnost2 * rychlost2.x *
rychlost2.x).ToString();

txt = GameObject.Find("hyb").GetComponent<Text>();
txt.text    =    (hmotnost1    *    rychlost1.x    +    hmotnost2    *
rychlost2.x).ToString();
```

4.3 Úloha 2 – Dokonale nepružná srážka koulí

Dvě koule o hmotnosti m_1 a m_2 se pohybují proti sobě po stejné přímce a srazí se. Srážka je dokonale nepružná, tj. koule po srážce od sebe „neodskočí“ a pohybují se dále společně. Kinetická energie E_1 první koule je před srážkou dvacetkrát větší než kinetická energie E_2 druhé koule. Je možné, aby se koule po srážce pohybovaly ve směru pohybu koule s menší kinetickou energií? Jaká podmínka pro to musí být splněna? (Řešené úlohy, 2006)

4.3.1 Řešení úlohy

Pro řešení je potřeba znalostí střední školy a využití zákona zachování hybnosti. Úloha je řešena v izolované soustavě, což znamená, že celková hybnost této soustavy se zachovává (součet hybností těles v této soustavě je konstantní).

Označme tedy:

m_1 … jako hmotnost první koule,

m_2 … jako hmotnost druhé koule,

$\overrightarrow{v_1}$ … jako počáteční rychlost první koule,

$\overrightarrow{v_2}$ … jako počáteční rychlost druhé koule,

$\vec{v}$ … jako výslednou rychlost obou koulí,

E_1 … jako kinetickou energii první koule,

E_2 … jako kinetickou energii druhé koule,

$\overrightarrow{p_1}$ … jako hybnost koule o hmotnosti m_1,

$\overrightarrow{p_2}$ … jako hybnost koule o hmotnosti m_2,

$\vec{p}$ … jako celkovou hybnost soustavy.

Ze zákona zachování hybnosti:

$$\overrightarrow{p_1} + \overrightarrow{p_2} = \vec{p},$$

$$m_1\overrightarrow{v_1} + m_2\overrightarrow{v_2} = (m_1 + m_2)\vec{v}.$$

Vzhledem k tomu, že se obě koule pohybují po přímce, můžeme přepsat tuto rovnici do skalárního tvaru. Pro řešení úlohy pak bude vhodné osu x orientovat ve směru pohybu koule s kinetickou energií E_2. Upravením tedy dostaneme tvar:

$$-m_1v_1 + m_2v_2 = (m_1 + m_2)v.$$

Aby dle zadání došlo k pohybu ve směru koule s kinetickou energií E_2, musí být splněna podmínka:

$$m_1v_1 < m_2v_2.$$

Kinetickou energii koule o hmotnosti m_1 lze zapsat ve tvaru

$$E_1 = \frac{1}{2}m_1{v_1}^2,$$

kinetickou energii koule o hmotnosti m_2 pak ve tvaru

$$E_2 = \frac{1}{2}m_2{v_2}^2.$$

Z obou tvarů vyjádříme rychlosti v_1 a v_2:

$$v_1 = \sqrt{\frac{2E_1}{m_1}},$$

$$v_2 = \sqrt{\frac{2E_2}{m_2}}.$$

Tyto vztahy pro rychlosti využijeme a dosadíme je do výše zapsané podmínky, čímž dostaneme:

$$m_1\sqrt{\frac{2E_1}{m_1}} < m_2\sqrt{\frac{2E_2}{m_2}}.$$

Umocněním tohoto vztahu získáme nerovnici:

$${m_1}^2\frac{2E_1}{m_1} < {m_2}^2\frac{2E_2}{m_2}.$$

Nyní zbývá jen upravit tuto nerovnici do vhodnějšího vztahu, ze kterého budeme schopni popsat výsledek úlohy. To zajistíme vydělením celé nerovnice dvěma, zkrácením příslušným hmotností. Pro lepší přehlednost pak upravíme nerovnici do podílového tvaru, čímž dostaneme:

$$\frac{E_1}{E_2} < \frac{m_2}{m_1}.$$

Ze zadání víme, že energie E_1 je dvacetkrát větší než energie E_2, takže do podmínky můžeme dosadit za podíl $\frac{E_1}{E_2}$ hodnotu 20:

$$20 < \frac{m_2}{m_1}.$$

Vynásobením nerovnice hmotností m_1 získáme podmínku pro řešení ve tvaru:

$$20m_1 < m_2.$$

Společný pohyb koulí ve směru koule, která má dvacetkrát menší kinetickou energii, je možný. Musí být splněna podmínka, ve které hmotnost koule s menší kinetickou energií bude alespoň dvacetkrát větší než hmotnost koule s větší kinetickou energií.

4.3.2 Modelace v Unity

Úloha 3 je podobná úloze 2. Budeme tedy vycházet ze stejné scény a stejného zdrojového kódu jako v předchozí úloze. Zadání je ale zaměřené tentokrát na nepružnou srážku, proto bude nutné doplnit podmínku, ve kterém odlišíme pružnou a nepružnou srážku. Na scénu

jsme proto umístili tlačítko, kterým můžeme přepínat mezi typem srážky. Dále je do scriptu potřeba výslednou rychlost koulí po nepružné srážce.

Podmínka se bude týkat zaškrtnutého, respektive nezaškrtnutého tlačítka. Pokud bude tlačítko zaškrtnuté, bude se jednat o pružnou srážku, v opačném případě se bude jednat o srážkou nepružnou. Pro druhou možnost doplníme výslednou rychlost, kterou lze vypočítat jako součet hybnosti první a druhé koule vydělený součtem hmotností obou objektů.

Jelikož se obě koule „spojí“, budou obě rychlosti shodné, rychlost první koule po srážce se bude rovnat rychlosti druhé koule po srážce.

```
if (GameObject.Find("pruzn").GetComponent<Toggle>().isOn)
{
v1new = (hmotnost1 * v1 + hmotnost2 * (2 * v2 - v1)) / (hmotnost1 +
hmotnost2);
v2new = (hmotnost2 * v2 + hmotnost1 * (2 * v1 - v2)) / (hmotnost1 +
hmotnost2);
}
else
{
v1new = (hmotnost1*rychlost1.x +
hmotnost2*rychlost2.x)/(hmotnost1+hmotnost2);
UnityEngine.Debug.Log(v1new);
v2new = v1new;
}
```

5 Zařazení počítačových modelů vybraných úloh při výuce fyziky

5.1 Modely ve výuce

Fyzika je jedním z předmětů, ve kterém je důležité, aby si žáci uměli danou problematiku představit. K ověřování teoretických základů jsou totiž často používány metody pozorovací nebo experimentální. Součástí výuky fyziky na některých školách jsou pak také laboratorní práce. Modely jsou nedílnou součástí vyučování fyziky, můžeme je považovat za nástroj, kterým může učitel demonstrovat svým žákům konkrétní fyzikální jev.

Jak již bylo zmíněno v kapitole 3.1, pod pojmem model si představíme zjednodušení reálné situace se zachováním stěžejních vlastností zkoumaného objektu nebo jevu. Při výuce se tedy naskytuje možnost vytvářet takové situace, ve kterých lze přecházet od konkrétních příkladů k obecným závěrům nebo naopak. (Válek, 2014, str. 22-23)

Je důležité, aby se žáci nezaměřovali pouze na poznání modelu, ale dokázali propojit tento model s fyzikální podstatou.

5.2 Klasifikace modelů

Modely lze dělit do různých skupin podle zvolených kritérií, mezi které spadá například obsah, cíl nebo zaměření. Záleží jen na autorovi, které kritérium zvolí. Jedním z takových dělení je dělení podle didaktických potřeb v podání Rakušana H. Mucke.

Ten rozděluje modely ve vyučování fyziky na modifikující, transformační, simulační, ilustrující a formalizující.

Pod modifikujícími modely si můžeme představit především změnu v měřítku. Výsledky získané experimentem lze pak převést na reálnou situaci.

Transformační modely, jak napovídá název, převádějí reálnou situaci na souhlasný systém, pomocí kterého se tuto situaci snažíme objasnit co nejjednodušším způsobem. Nelze ovšem obecně dělat úplné závěry o reálné situaci.

Simulační modely jsou známé také pod názvem imitující modely. Základem těchto modelů je vnější napodobení originálu. Odlišnost od reálné situace nastává v činnosti modelu, která se řídí vlastními zákonitostmi.

Obrazné znázornění originálního jevu je známé jako ilustrující model. Asociace zákonů uživatelem probíhá pozorováním pracovní a ilustrující funkce.

Posledním typem jsou formalizující modely, ve kterých jsou reálné situace popisovány pomocí matematických znakových systémů. (Vachek, 1980, str. 16)

5.3 Zařazení modelů do výuky

Úlohy, které byly popsané výše, byly vybrány především na základě problematických míst výuky fyziky, se kterými se mohou setkat učitelé během praxe. Další faktorem, podle kterého byly úlohy vyselektovány, byla souvztažnost s mechanikou. Fyziku, jak je známo, lze rozdělit do mnoha oblastí. V této práci jsem se zaměřil na oblast mechaniky, která je velmi rozsáhlá a pod kterou spadá také několik kapitol. Je vhodné dodat, že se jedná o subjektivní pohled na danou věc.

Problematické situace se týkají především slovních úloh zaměřených na pohyb, na které mohou kantoři narazit také při výuce matematiky. Tyto úlohy nemívají vysokou míru úspěšnosti v oblasti řešení. Důvodů může být několik, velmi často žáci nepochopí samotné zadání úkolu, popřípadě nedokážou sestavit rovnici nebo soustavu rovnic, pomocí které by řešení vypočítali. Vytvořený model by tedy mohl usnadnit žákům pochopení zadání úlohy a tím i sestavení rovnice nebo soustavy rovnic.

Zbylé dvě úlohy (úloha 2 a 3) byly vybrány z důvodu vhodnosti modelace. Jak pružná, tak i nepružná srážka jsou ideálními tématy, ve kterých lze použít modely při výuce. V platformě Unity lze nastavovat různé vlastnosti jednotlivých hmotných bodů, které ovlivní výsledek úlohy.

Žáci osmých a devátých tříd během školního roku narazili na zmíněné učivo, které bylo vyučováno nejprve bez těchto modelů. Po dokončení jednotlivých kapitol vyplňovali žáci pracovní listy, do kterých byly zahrnuty příklady popsané v předchozích kapitolách. Následně probíhala výuka, při které byly využity tyto modely. Opět byla práce žáků s využitím pracovních listů reflektována. Výsledky jednotlivých úloh pak byly porovnány.

5.4 Úloha 1 – model ve výuce

Úlohy číslo jedna a dva se týkají učiva, které není na základní škole probírané ve fyzice, ale lze se s ním setkat v jiném předmětu. V matematice se s tématem slovní úlohy o pohybu setkají učitelé většinou v průběhu osmého ročníku, možností je také případná výuka tohoto tématu na začátku devátého ročníku. První situace se týkala také základní školy, ve které byly modely použity.

5.4.1 Popis tříd

V obou třídách mělo být dohromady 43 žáků, nicméně skutečný stav byl nepatrně odlišný z důvodu absence některých z nich. Pracovní list tedy vyplnilo 38 žáků. Ve třídě 8. B je celkově 21 žáků, z toho 12 chlapců a 9 dívek. Tato třída pracuje v hodinách fyziky velmi dobře, většina žáků má o učivo zájem, což dokazují nejrůznějšími dotazy v průběhu vyučovací hodiny. Zapojují se často také do diskuse. Aktivnější jsou především hoši. V této třídě jsou také 3 chlapci se speciálními vzdělávacími potřebami. Z vědomostního hlediska považuji zmíněnou třídu za průměrnou.

Do třídy 8. C dochází celkem 21 žáků, z toho 12 chlapců a 9 dívek. Přestože je tato třída složená stejně jako předchozí, pracuje se v této třídě o něco lépe. Žáci mají rádi hodiny, ve kterých mohou něco zkoumat a prohlubovat tak své znalosti. Nový program, ve kterém lze využít aplikace k simulaci reálných problémů, byl pro ně motivátorem a z mého pohledu pracovali o něco lépe než druhá třída. Vědomostně řadím žáky třídy 8. C do nadprůměru, což dokládá vhodná argumentace při hodinách a lepší průměrné hodnocení. V této třídě je pak pět hochů se speciálními vzdělávacími potřebami.

V obou třídách se velmi dobře pracuje, žáci spolupracují a jsou aktivní. Obrovskou výhodou je jejich vzájemná komunikace, a to zejména v případech, kdy někteří žáci nerozumí zadanému učivu. Dalším obrovským plusem při výuce v těchto třídách je, že se žáci nebojí odpovídat na zadané otázky, i kdyby to znamenalo, že odpoví chybně.

5.4.2 Popis hodiny

Vyučovací hodina proběhla ve dvou osmých ročnících na přelomu května a června. Hlavním cílem těchto vyučovacích hodin bylo, aby žáci vypočítali slovní úlohy zaměřené na pohyb. Žáci by z dřívějších hodin měli umět sestavovat a počítat rovnice, měli by také vypočítat soustavu dvou rovnic o dvou neznámých a v neposlední řadě pracovat se vzorcem $v = \frac{s}{t}$.

Jednalo se o hodinu, která navazuje na téma slovní úlohy o pohybu. Tyto úlohy byly řešené pomocí rovnic. Žáci zatím neumějí používat soustavu rovnic k řešení slovních úloh o pohybu, budou se tedy muset při výpočtu obejít bez nich. Nicméně měli by už umět pracovat s rovnicemi, vyjadřovat z rovnic neznámou, sestavovat výrazy na základě zadání úlohy a vytvořit rovnici, kterou lze slovní úlohu vyřešit.

Obě vyučovací hodiny probíhaly v odborné učebně fyziky, ve které je k dispozici dataprojektor. Na tomto přístroji pak byla promítána animace vytvořená v platformě Unity.

5.4.2.1 Struktura hodiny

V úvodu hodiny nastínil učitel téma, které budou žáci v hodině probírat a popsal, jakým způsobem bude hodina probíhat. Poté dostali žáci dvě úlohy, jejichž zadání bylo napsáno na papíru, který si nalepili do sešitu. Na vyřešení těchto úloh byla časová dotace 5 minut, které učitel využil k zápisu do třídní knihy a přípravě modelů.

Zadání úloh:

1. Průměrná rychlost běžce na lyžích, který byl na celodenním výletu, je $13\ \mathrm{km/h}$.
    a. Jakou vzdálenost ujede za 1 hodinu?
    b. Jakou vzdálenost ujede za 3 hodiny?
    c. Jakou vzdálenost ujede za 5 hodin?
    d. Jakou vzdálenost ujede za t hodin?
2. Podle jakého vzorce lze vypočítat dráhu pohybu běžce? Odvoď z tohoto vzorce vztah pro rychlost a čas. Jaká podmínka musí být splněna, abychom tento vzorec mohli použít?

Jejich úkolem bylo pracovat na těchto úlohách a zopakovat si tím řešení problematiky týkající se pohybu, kterou se zabývali v předchozích hodinách. Zároveň museli pracovat se vzorcem $v = \frac{s}{t}$, který je součástí učiva fyziky v sedmém ročníku základní školy.

V průběhu hodiny dostali žáci dvě úlohy o pohybu, které měli vyřešit. Ty samé úlohy dostali žáci také po několikanásobném shlédnutí animací v platformě Unity na začátku příští hodiny. Výsledky těchto úloh pak byly porovnány. Na výpočet obou úloh měli žáci 15 minut, což znamená, že zbývalo 25 minut z celkového času vyučovací hodiny. Cílem této hodiny bylo, aby žáci zvolili vhodný postup při řešení úloh o pohybu.

Přibližně 10 minut z hodiny byl využit k práci s animací k úloze jedna. Zadána byla úloha jedna, která je podrobně popsána v kapitole 5. Žákům nebylo nastíněno řešení úlohy, jen měli možnost vidět zadání úlohy pomocí platformy Unity. Několikrát viděli animaci, kterou se následně pokusili využít k řešení. Po vypršení časové limitu zbývalo přibližně 10 minut. Závěrečná část pak byla věnována hodnocení, kdy žáci dostali několik otázek a vybírali jednu z nabízených možností. Tyto otázky jsou popsány v další kapitole.

5.4.3 Reflexe hodiny

Vyučovací hodiny v obou třídách měly vysoké pracovní tempo. Aktivní byli především žáci, které se dle mého názoru podařilo zaujmout. První motivační úloha pro většinu nepředstavovala velký problém. Žáci se bez otázek pustili do řešení. Druhý úkol už byl obtížnější, poněvadž někteří žáci nepochopili, co se po nich chce. Nedokázali tak zapsat vztah mezi rychlostí, dráhou a časem.

Po společné kontrole následovala samostatná práce, při které byla zadána úloha k řešení. Žáci obou tříd znali úlohu spíše z matematického hlediska, někteří z nich se proto doptávali na pojmy, se kterými se nesetkali. Jiní se obraceli s prosbou o pomoc, jelikož si nebyli jistí, zda správně pochopili zadání úlohy. Větší část se ale pustila do řešení. Výsledky jednotlivých úloh budou rozebrány v další kapitole.

Z kázeňského hlediska se ani v jedné hodině nevyskytl problém, který by ovlivnil nebo narušil chod hodiny. Žáci tak měli dostatečný prostor k řešení zadaných úloh.

6 Shrnutí získaných poznatků z výuky

V následující části je podrobněji popsáno šetření, které sloužilo jako nástroj k hodnocení výuky fyziky s využitím počítačových modelů. Realizované šetření využívalo dotazníku, ke kterému budou v další části uvedeny hypotézy a výsledky.

6.1 Cíle výzkumu

Cílem výzkumu bylo zjištění, zda vytvořené modely k úlohám pomáhají žákům při řešení úloh o pohybu a také jak žáci vnímají platformu Unity. Proto byly stanoveny následující dílčí výzkumné cíle:

1. Porozumění zadání úlohy jedna s využitím vytvořeného modelu.
2. Přehlednost úlohy v platformě Unity z pohledu žáků.

Využitím dotazníku byly testovány zmíněné cíle a vyhodnoceny jejich výsledky.

6.2 Výzkumný soubor

Základní soubor tvořili žáci dvou tříd osmého ročníku základní školy, na které vyučuji obě zmíněné třídy. Celkový počet respondentů ve věkovém rozmezí $13 - 15$ let je 38. Sběr dat dotazníku probíhal v květnu 2022 pouze v papírové podobě. Dotazník vyplnilo celkem 22 chlapců (58%) a 16 dívek (42%), kteří měli 10 minut na vyhodnocení. V těchto třídách je celkem devět žáků se speciálními vzdělávacími požadavky na vyučování. Je důležité upozornit na fakt, že mezi těmito žáky jsou pouze hoši.

6.3 Výzkumné otázky

Na základě zmíněných výzkumných cílů byly vytvořeny výzkumné otázky a hypotézy. Zjišťovány byly pak četnosti jednotlivých odpovědí. Poté byly testovány hypotézy na základě těchto výsledků.

Otázky:

1. Jsem:
    a. Kluk b. Holka
2. Zadání úlohy jedna mi přišlo srozumitelné.
    a. Rozhodně ano b. Spíše ano c. Spíše ne d. Vůbec ne
3. Využití aplikace v Unity mi pomohlo porozumět zadání úlohy jedna.
    a. Rozhodně ano b. Spíše ano c. Moc ne d. Vůbec ne
4. Úloha jedna byla v aplikaci přehledná.
    a. Rozhodně ano b. Spíše ano c. Spíše ne d. Vůbec ne

6.4 Hypotézy výzkumného šetření

Hypotézy byly sestavovány na základě zkušeností učitelů z výuky fyziky a jejich pohledů na danou problematiku.

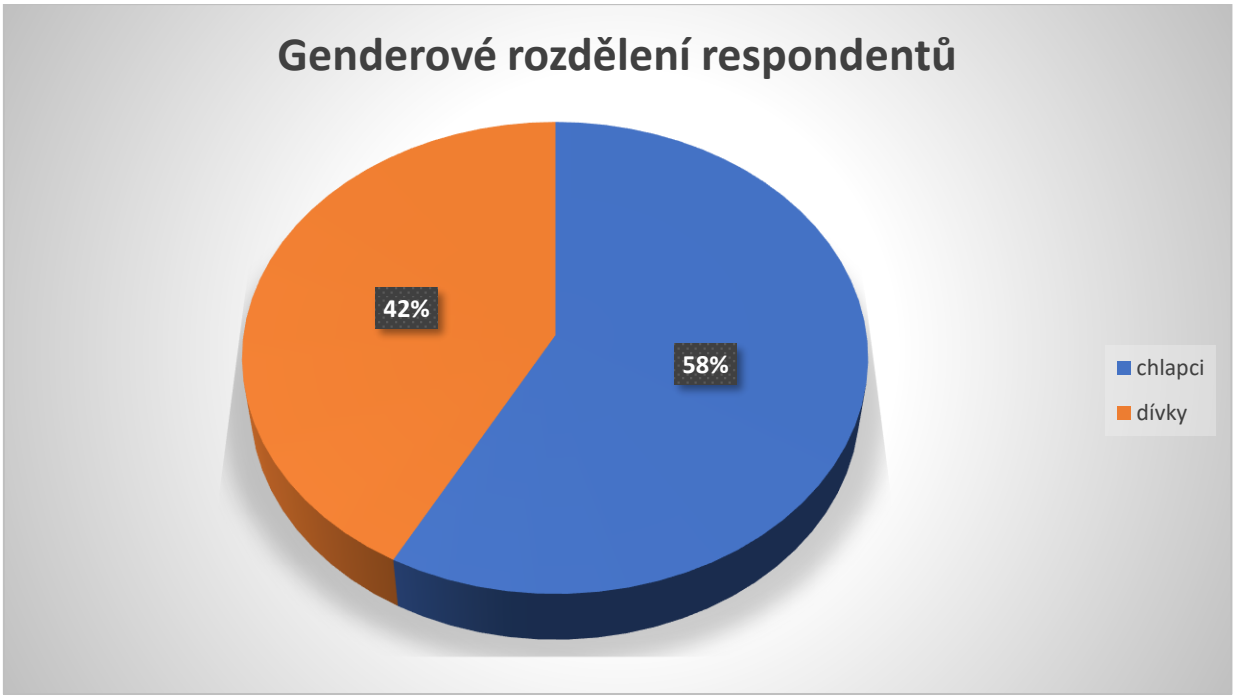
Hypotézy:

1. Více jak osmdesáti procentům žáků osmého ročníku pomohlo využití platformy Unity k pochopení zadaní úlohy jedna.
2. Dívkám v obou třídách pomohlo využití modelů více než chlapcům.

6.5 Výsledky

6.5.1 Výsledky otázky č. 1

První otázka měla za cíl zjistit, jestli je rozdíl ve vnímání modelů jednotlivých úloh mezi chlapci a děvčaty. Její výsledky totiž mohou hrát roli ve vnímání animací vytvořených v různých programech. Užitečná pak byla zejména k potvrzení nebo vyvrácení druhé hypotézy, která srovnávala dopady animace na řešení slovní úlohy žáky.



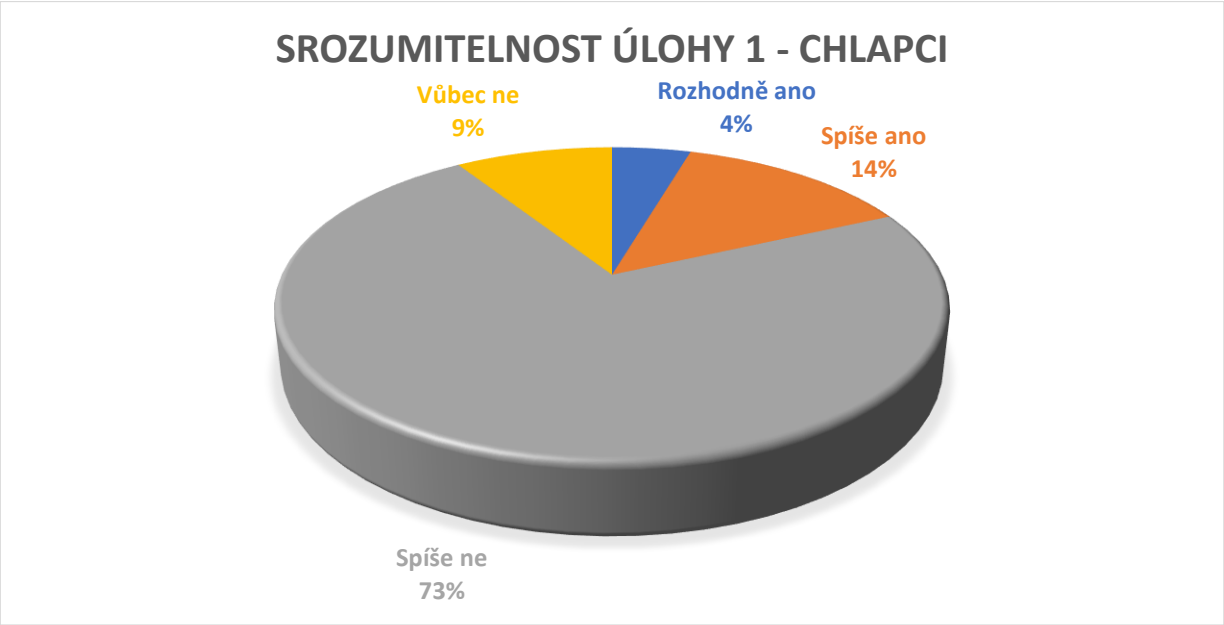
Graf 1: Genderové rozdělení respondentů

Z celkového počtu čtyřiceti tří žáků vyplnilo dotazník 22 chlapců (58 % z celkového počtu respondentů) a 16 dívek (42 % z celkového počtu respondentů).

6.5.2 Výsledky otázky č. 2

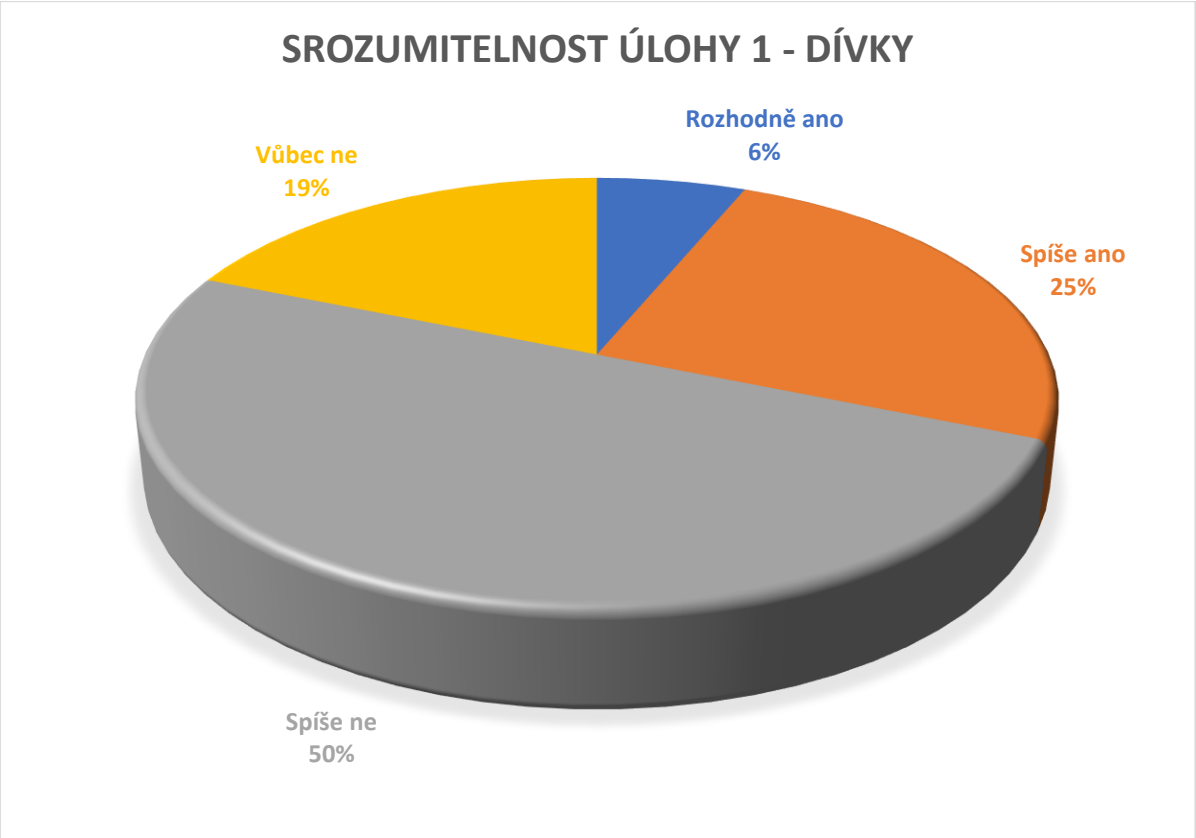
Druhá otázka byla zaměřena na subjektivní vnímání zadání první úlohy. Jejím cílem bylo ověřit, zda respondenti porozuměli zadání úlohy, tedy jestli byl slovní popis problému dostatečně srozumitelný a výstižný. Jednalo se o slovní úlohu zaměřenou na pohyb hmotných bodů ve směru za sebou a proti sobě.

Po zodpovězení této otázky byla pozornost zaměřena pouze na žáky, kterým nebyla úloha srozumitelná před shlédnutím animace v platformě Unity. Žáci, kteří pochopili zadání bez shlédnutí aplikace, jsme dále nezahrnovali do výsledků šetření.



Graf 2: Srozumitelnost úlohy 1 – chlapci

Data získaná zodpovězením otázky 2 byla rozdělena z hlediska pohlaví respondentů. Chlapci nejčastěji kroužkovali odpověď spíše ne, tedy spíše nerozumím zadání úlohy (16 respondentů, což je 73 % z celkového počtu chlapců). Druhá nejčastěji uváděná odpověď byla spíše ano, žáci tedy rozuměli zadání, zřejmě si ale nebyli dostatečně jistí (3 chlapci, což je 14 % z celkového počtu chlapců). Úloze vůbec nerozuměli dva žáci (9 % z celkového počtu dotazovaných žáků) a pouze jednomu žákovi přišla úloha 1 zcela srozumitelná (4 % z celkového počtu dotazovaných žáků).

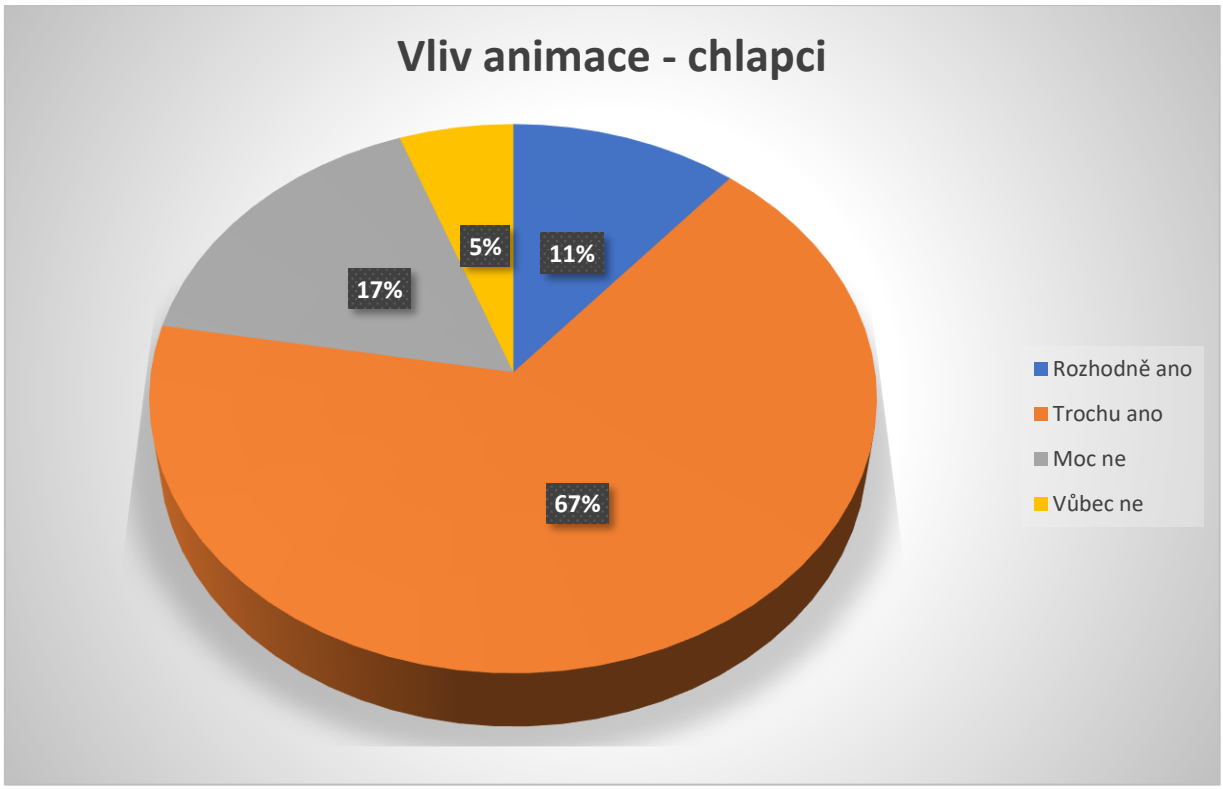


Graf 3: Srozumitelnost úlohy 1 - dívky

Nejčastější uváděná odpověď na otázku číslo 2 u dívek byla stejná jako u chlapců, tedy spíše ne. Tuhle odpověď zakroužkovalo 8 dívek (50 % ze všech dotazovaných dívek). Druhá nejčastěji označovaná odpověď byla stejná jako u chlapců, 4 dívky (25 % z dotazovaných respondentek) uvedly, že zadání spíše rozuměly. Tři dívky (19 % všech dotazovaných dívek) zvolily možnost vůbec ne a stejně jako u chlapců pouze jedna dívka (6 % ze všech dotazovaných dívek) slovnímu zadání zcela porozuměla.

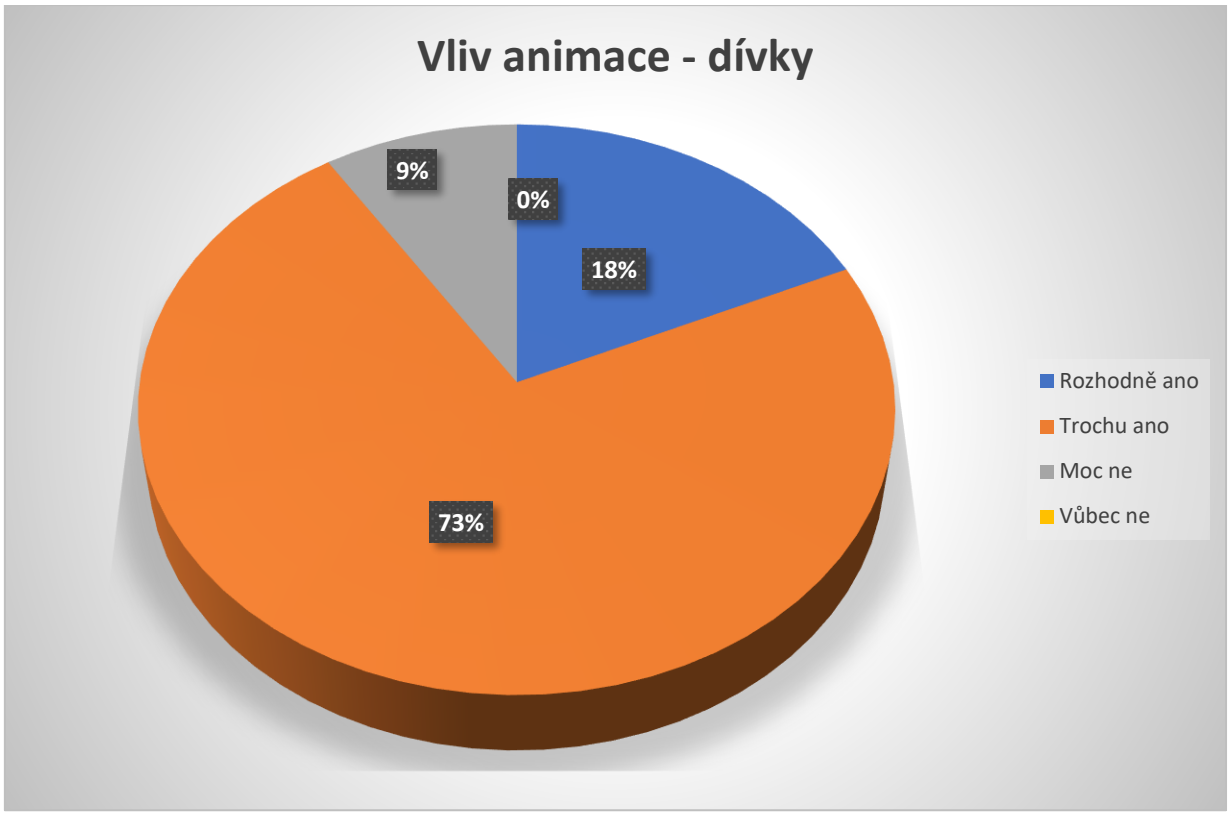
6.5.3 Výsledky otázky č. 3

Třetí otázka byla zaměřena na samotnou animaci vytvořenou v platformě Unity. Jejím cílem bylo vyhodnotit, zda tato animace pomohla respondentům v pochopení zadání úlohy. Žáci mohli opět zvolit jednu ze čtyř možností: rozhodně ano; trochu ano; moc ne; vůbec ne. Data získaná z odpovědí na tuto otázku jsou stěžejní pro vyhodnocení první hypotézy. Odpovědi pak byly opět rozděleny podle pohlaví respondentů.



Graf 4: Vliv animace (úloha 1) – chlapci

Nejčastější možnost, kterou chlapci označili, byla trochu ano. Celkem čtrnácti žákům (67 % všech dotazovaných chlapců) pomohlo shlédnutí animace v aplikaci Unity k pochopení zadání úlohy. Jedná se opět o subjektivní pohled, při kterém mohou mít někteří žáci pocit, že zadání úlohy na základě aplikace pochopili. Druhou nejčastěji uváděnou odpovědí (2 žáci, tedy 11 % z celkového počtu dotazovaných chlapců) bylo, že hochům využití animace ve výuce pomohlo k pochopení zadání úlohy. Tři z dotazovaných respondentů (17 % z celkového počtu) uvedli, že jim aplikace moc nepomohla. Jednomu z žáků pak nepomohla vůbec (5 %).



Graf 5: Vliv animace (úloha 1) – dívky

U dívek byly odpovědi na otázky téměř shodné. Z celkového počtu dvanácti dívek uvedlo 73 % z nich, že jim animace trochu pomohla. Dvě dívky (18 % z celkového počtu respondentek) zvolily možnost rozhodně ano. Rozdíl byl v tom, že pouze jedna dívka (9 % z celkového počtu dotazovaných dívek) zakroužkovala možnost, že jí aplikace moc nepomohla v pochopení úlohy. Ani jedna dívka nezvolila možnost vůbec ne.

6.5.4 Výsledky otázky č. 4

Poslední otázka nesloužila k potvrzení či vyvrácení hypotéz. Jejím cílem bylo zjistit, jak žáci vnímají prostředí, ve kterém byla tato aplikace vytvářena. Pomohla posoudit, jestli byla úloha namodelována dostatečně vzhledem k zadání příkladu.

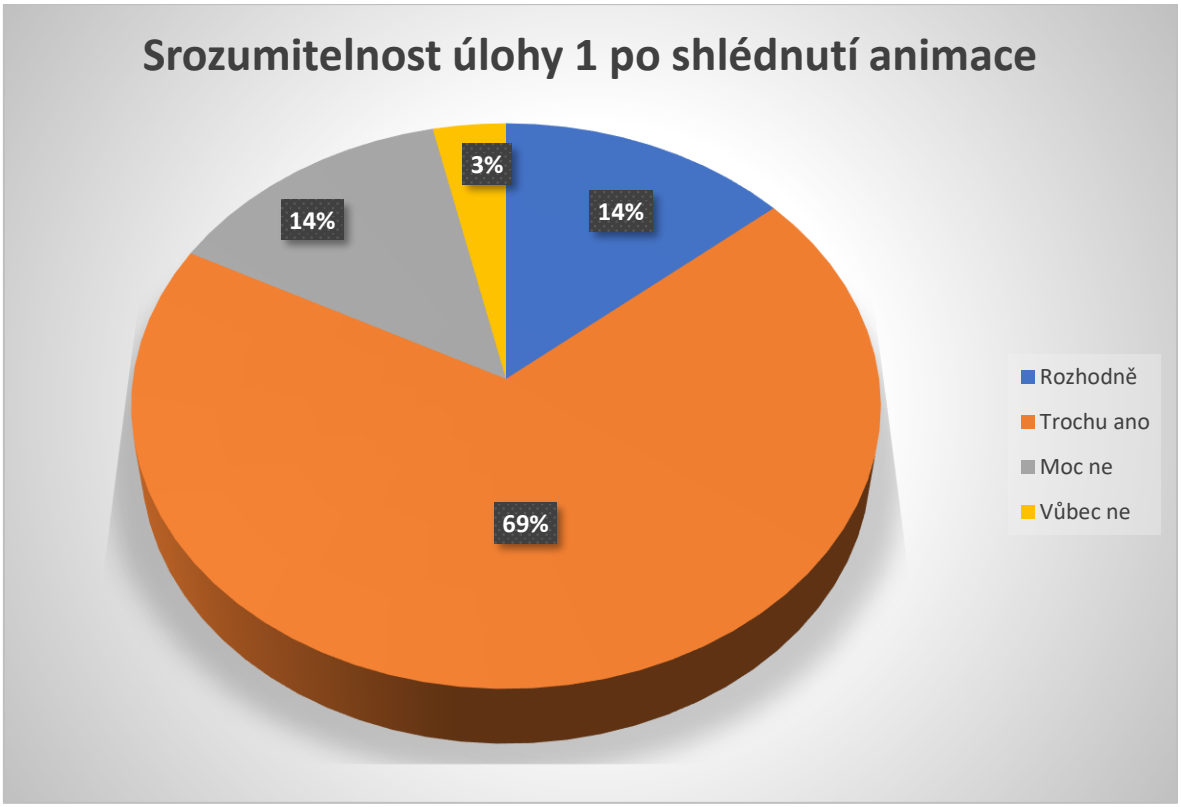
Graf 6: Přehlednost úlohy jedna v aplikaci

Výsledky byly poměrně jasné. Většině žáků přišla aplikace méně nebo více přehledná. Obě varianty zvolilo celkem 36 z 38 respondentů, což je celkem 95 %. 23 žáků uvedlo, že jim aplikace přišla spíše přehledná. Pouze dvěma dotazovaným (5 % všech žáků) přišla animace v aplikaci spíše nepřehledná.

6.6 Výsledky hypotéz

6.6.1 Diskuse výsledků hypotézy H1

První hypotéza se týkala pomoci animace vytvořené v platformě Unity v pochopení zadání úlohy číslo 1. Využitím výsledků z výzkumného řešení, můžeme hypotézu H1 potvrdit či vyvrátit. Pro konečný verdikt byly zkoumány odpovědi žáků, kteří před shlédnutím animace uvedli, že nerozumí zadání úlohy.



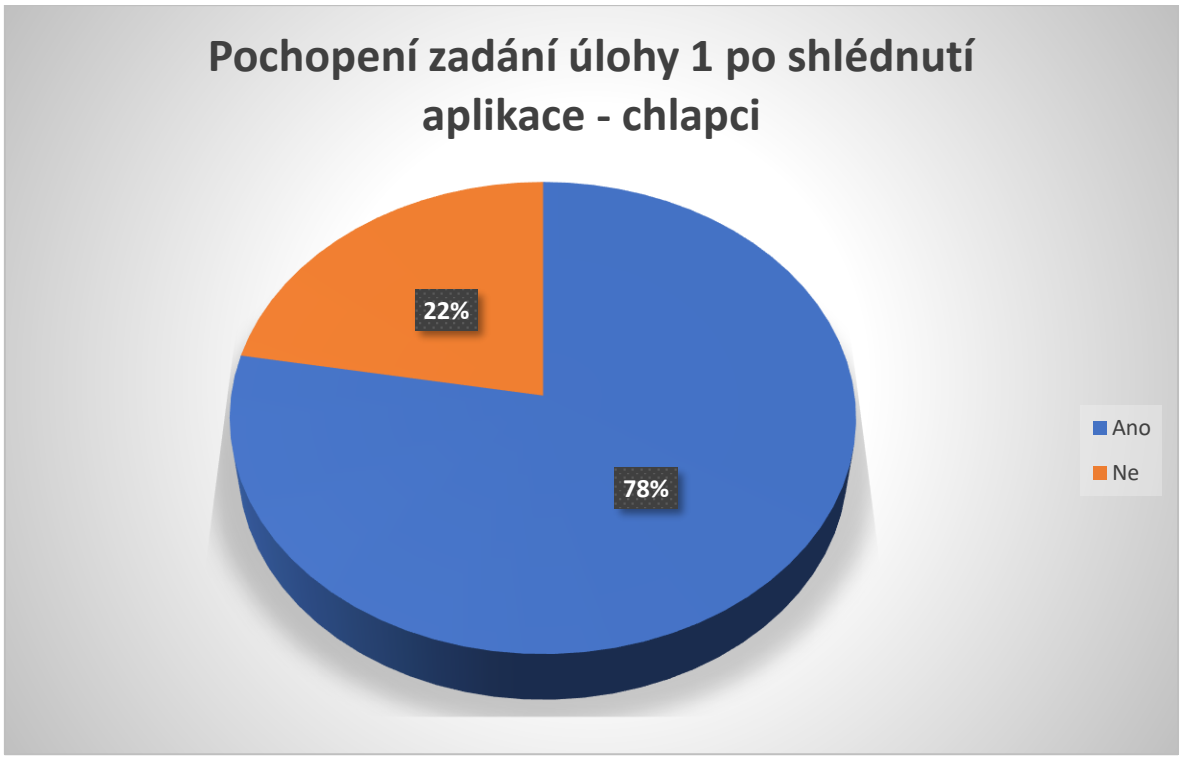
Graf 7: Srozumitelnost úlohy 1 po shlédnutí animace

Z tohoto počtu celkem 83 % chlapců a dívek uvedlo, že jim úloha přišla srozumitelná po shlédnutí animace vytvořené v platformě Unity. Hypotézu H1 tedy můžeme potvrdit. Přispívá k tomu i fakt, že tito žáci správně zakreslili danou situaci, která byla součástí rozboru úlohy.

Většina žáků i přesto nedokázala zadanou úlohu vypočítat. Vliv na tuto skutečnost může mít například nedostatečný matematický aparát, numerická chyba, špatná interpretace výsledků a jiné. Neznamená to však, že žáci zadané úloze nerozumí.

6.6.2 Diskuse výsledků hypotézy H2

Dívkám v obou třídách pomohlo využití modelů více než chlapcům. Tak zněla hypotéza H2. Potvrdit nebo vyvrátit ji můžeme porovnáním žáků a žákyň, kteří úlohu zprvu nechápali a po využití aplikace jim úloha přišla srozumitelná.



Graf 8: Pochopení zadání úlohy 1 po shlédnutí aplikace – chlapci

Graf 9: Pochopení zadání úlohy 1 po shlédnutí aplikace – dívky

Z výše uvedených grafů je zřejmé, že hypotézu H2 můžeme potvrdit. Pouze jedné dívce z celkového počtu jedenácti dívek nepomohla aplikace v pochopení úlohy. U chlapců byl tento počet daleko větší, z toho důvodu nemůžeme hypotézu vyvrátit. Je vhodné ovšem zmínit, že někteří z hochů mohli být žáci se speciálními vzdělávacími požadavky. To také mohlo ovlivnit získané hodnoty.

Závěr

Hlavním cílem diplomové práce bylo vytvořit počítačový model k vybraným úlohám z mechaniky. Byly zvoleny tři úlohy, ke kterým byly vytvořeny celkem dva programy. První animace znázorňuje úlohu, ve které se dva hmotné body pohybují podle zadání za sebou nebo proti sobě. Aplikace umožňuje vkládat různé rychlosti, kterými se mají objekty pohybovat. Pokud bychom chtěli měnit vzdálenost, mohli bychom přepsat souřadnice bodů v samotném scriptu nebo přidat na scénu další vstupní pole, které by změnu souřadnic umožňovalo. Úloha byla vybrána na základě subjektivního pocitu z řešení úloh o pohybu žáky na základní škole. Toto téma se pak nejčastěji vyučuje v matematice v osmém nebo devátém ročníku, vyučováno může být také na střední škole v rámci fyziky.

Druhá animace je zaměřena spíše pro žáky středních škol, využita by mohla být i na základní škole. Znázorňuje pružnou i nepružnou srážku dvou těles a jejich výsledný pohyb. Její struktura vychází z první úlohy, je ovšem doplněna o další „vychytávky“. Uživatel může na vstupu zadat hmotnosti a rychlosti obou těles a platforma Unity pak počítá hybnost a energii před srážkou i po srážce v závislosti na typu srážky.

Tvorba těchto animací byla rozepsána do dílčích kroků, které mají za úkol přiblížit čtenáři, jakým způsobem lze v platformě Unity vytvořit program. Diplomová práce obsahuje také samotné řešení úloh, které je nezbytné pro zvolení vhodné scény, ale hlavně pro ověření výsledků, které získáme z programu.

Teoretická část pak zahrnuje popis vývoje mechaniky včetně důležitých milníků, které byly základem pro další zkoumání. Díky poznatkům z kvantové mechaniky docházelo mimo jiné k rozvoji elektroniky. Jsou to právě počítače, které umožnili vytvářet různé simulace a modely a přibližovat tak představu o různých dějích a situacích. V diplomové práce je také stručně popsán proces tvorby počítačových modelů. Za zmínku stojí také popis některých programů, které byly nebo stále jsou využívány k modelování úloh. Podrobněji je pak popsána funkce platformy Unity, ve které byly úlohy vytvářeny.

Vedlejším cílem této práce bylo otestovat tyto animace v praxi. Otestována byla pouze první úloha, která byla zařazena do výuky fyziky pro žáky osmého ročníku, ač se jedná, jak již bylo zmíněno, o slovní úlohy o pohybu, které jsou vyučovány v matematice.

Žáci měli možnost sledovat práci s aplikací, která jim měla přiblížit zadání úlohy a která jim měla pomoci při řešení této úlohy. Šetření proběhlo dotazníkovou formou, kde žáci odpovídali na čtyři předem stanovené otázky. Ze získaných dat byly potvrzeny hypotézy. Více než osmdesáti procentům žáků pomohla, podle jejich hodnocení, animace vytvořená v platformě Unity k pochopení zadání úlohy, což se u některých respondentů projevilo také při opětovném řešení této úlohy následující hodinu. Zároveň výzkum potvrdil, že animace pomohla více dívkám než chlapcům. V práci je pak podrobněji rozepsáno samotné šetření i složení respondentů.

Hlavního cíle práce bylo dosaženo, animace bychom mohli i nadále upravovat nebo vytvářet nové. Vytvořené animace by pak mohli žákům pomoci během studia na základní či střední škole, a to zejména v oblasti abstraktních věd. Výzkumem se podařilo ověřit, že aplikace měla velký vliv na pochopení zadání úlohy a pozitivně ovlivnila také řešení některých respondentů. Tato práce by mohla sloužit ke snazšímu vytváření programů v platformě Unity zejména v oblasti mechaniky.

Resumé

Diplomová práce popisuje tvorbu animace vybraných úloh z mechaniky v platformě Unity. V teoretické části práce je popisován vývoj mechaniky, díky které se rozvíjely technologie, mezi které se řadí i počítače. V této části jsou dále vymezeny pojmy model a modelace a popisovány některé programy, které byly nebo jsou využívány ve výuce.

Praktická část se věnuje platformě Unity a tvorbě animací. Byly vytvořeny dvě animace k třem vybraným úlohám z mechaniky, které jsou řešeny v rámci základních a středních škol. K funkčnosti aplikace jsou potřeba příkazy, které jsou v této části rovněž popsány. V závěrečné pasáži diplomové práce je prostřednictvím dotazníkového šetření zjišťován vliv vytvořených aplikací na výuku fyziky v osmém ročníku základní školy. Bylo zjištěno, že aplikace měla vliv na pochopení i řešení úlohy a že dívkám pomohla aplikace k úloze o něco více než chlapcům.

Summary

The thesis describes a creation of animation of selected function from mechanics in Unity platform. The theoretical part is focused on development of the mechanics as a device, that led to progress of technologies that include computers. Terms model and modulation are defined in the theoretical part. Some programs that are used in teaching are also described in that part.

The practical part occupies with the Unity platform and the creation of animations. Two animations for three selected machanics tasks for primary a secondary education were made. Functionality of the applcations requires commands which are also described in this part. The conclusion of the thesis contains a questionnaire that finds out an influence of the created applications on the teaching of physics in the eighth grade of primary school. The result was that the application had an impact on the understanding and the solution of the task and that the application had a bigger influence on girls than on boys.

Seznam použité literatury a elektronických zdrojů

Použitá literatura

BORY, Pavel. *C# bez předchozích znalostí.* 1. vyd. Brno: Computer Press, 2016, 255 s. ISBN 978-80-251-4686-6.

ČERNÝ, M. *Počítačové simulace a modelování ve výuce fyziky v programu Algodoo.* MATEMATIKA-FYZIKA-INFORMATIKA, Praha: Prometheus, 2013, roč. 22, č. 3, s. 216-223. ISSN 1805-7705.

DVOŘÁK, L. *Famulus 3.5. Příručka uživatele*. Computer Equipment, Praha 1992, 310 s

FEYNMAN, Richard Phillips, Robert B. LEIGHTON a Matthew SANDS. *Feynmanovy přednášky z fyziky 1: revidované vydání s řešenými příklady*. 2. vyd. Přeložil Ivan ŠTOLL. Praha: Fragment, 2013. ISBN 978-80-253-1642-9.

HERNECK, Friedrich. *Průkopníci atomového věku*. Praha: Orbis, 1974. Stopy, fakta, svědectví (Orbis).

HOLAN, Tomáš. *Unity, první seznámení s tvorbou počítačových her.* Praha: CZ.NIC, 2020. ISBN 978-80-88168-60-7.

HORSKÝ, Jan, Jan NOVOTNÝ a Milan ŠTEFANÍK. *Mechanika ve fyzice*. Praha: Academia, 2001. ISBN 80-200-0208-1.

HÖSCHL, Cyril. *Eseje o mechanice*. Liberec: Technická univerzita, 2009. ISBN 978-80-7372-455-9.

KULHÁNEK, Petr. *Jak vznikal svět, aneb, Třináctero příběhů o kosmologii*. Praha: AGA, 2019. ISBN 978-80-906638-1-7.

LEPIL, Oldřich a Lukáš RICHTEREK. *Dynamické modelování*. Ostrava: Repronis, 2007. ISBN 978-80-7329-156-3.

MECHLOVÁ, Erika. *Vytváření fyzikálních pojmů u žáků*. Ostrava: Ostravská univerzita v Ostravě, 2014. ISBN 978-80-7464-358-3.

OPATRNÝ, Tomáš. *Základy moderní fyziky: studijní modul*. Olomouc: Univerzita Palackého v Olomouci, 2012. ISBN 978-80-244-3333-2.

SAMEK, Ladislav a František ČERNÝ. *Fyzika v příkladech*. Praha: Academia, 2014. Gerstner. ISBN 978-80-200-2319-3.

ŠTOLL, Ivan. *Dějiny fyziky*. Praha: Prometheus, 2009. ISBN 978-80-7196-375-2.

VACHEK, Jaroslav a Oldřich LEPIL. *Modelování a modely ve vyučování fyzice*. Praha: Státní pedagogické nakladatelství, 1980. Edice metodických příruček.

Elektronické zdroje

AXON, Samuel. Unity at 10: For better—or worse—game development has never been easier. *arstechnica.com* [online]. Ars technica, 27.09.2016. Dostupné z: https://arstechnica.com/gaming/2016/09/unity-at-10-for-better-or-worse-game-development-has-never-been-easier/

DRSKA, Ladislav. *Program Famulus.* In: Fakulta jaderná a fyzikálně inženýrská, České vysoké učení technické v Praze. [cit. 18.4.2022]. Dostupné z: http://www-troja.fjfi.cvut.cz/~drska/edu/webfyz/node23.html

GREGORCIC, BOR; BODIN, MADELEN. *Algodoo. A tool for encouraging creativity in physics teaching and learning.* In: The Physics Teacher, 55 (1), pp. 25–28, 2017. Dostupné z: http://www.algodoo.com/what-is-it/

JANEČEK, P. *Počítačový model jako moderní nástroj pro podporu výuky fyziky na základní a střední škole*. Olomouc, 2011. 109 s. Rigorózní práce. Univerzita Palackého.

Katedra didaktiky fyziky Matematicko-fyzikální fakulty UK [online]. KDF MFF UK: 2006. [Cit. 19.6.2022]. Dostupné z: http://reseneulohy.cz/cs/fyzika/mechanika.

VÁLEK, Jan. *Modelování fyzikálních jevů pro využití ve výuce fyziky na ZŠ a SŠ.* Olomouc, 2014. Disertační práce (Ph.D.). Univerzita Palackého v Olomouci, Přírodovědecká fakulta, Katedra experimentální fyziky, 2014. Dostupné z: https://theses.cz/id/8qkb0i/jan-valek disertacni-prace.pdf

Seznam obrázků

Seznam grafů

Seznam příloh

Příloha 1: Zdrojový kód k úloze 1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using System.Linq;

public class GameLogic : MonoBehaviour
{
    private IEnumerable<GameObject> objArray;
    //public static GameSystem Instance { get; private set; }
    GameObject koule1, koule2;
    Rigidbody2D component1, component2;
    bool beforeCrash = false;
    DateTime startTime = DateTime.Now;


    Vector2 rychlost1 = new Vector2(0, 0);
    Vector2 rychlost2 = new Vector2(0, 0);


    // Start is called before the first frame update
    void Start()
    {
        koule1 = GameObject.Find("koule1");
        component1 = koule1.GetComponent<Rigidbody2D>();

        koule2 = GameObject.Find("koule2");
        component2 = koule2.GetComponent<Rigidbody2D>();

    ResetButton();

        //this.ButtonPressed();
    }
    // Update is called once per frame

    void FixedUpdate()
    {

        #region texty

        var cas = GameObject.Find("time");
        Text txt = cas.GetComponent<Text>();
        double sec = (DateTime.Now - startTime).TotalSeconds;
```

```
        if (beforeCrash)
        {
            txt.text = startTime != DateTime.MinValue ?
sec.ToString() : "0";
            txt = GameObject.Find("draha1").GetComponent<Text>();
            txt.text = (rychlost1.x * sec).ToString();

            txt = GameObject.Find("draha2").GetComponent<Text>();
            txt.text = (rychlost2.x * sec).ToString();
        }
        #endregion

        component1.transform.position = new
Vector2(component1.transform.position.x + rychlost1.x *
Time.fixedDeltaTime, component1.transform.position.y + rychlost1.y
* Time.fixedDeltaTime);

        component2.transform.position = new
Vector2(component2.transform.position.x + rychlost2.x *
Time.fixedDeltaTime, 0);

        if (beforeCrash && (koule1.transform.position.x >
koule2.transform.position.x)) //srazka
        {
            beforeCrash = false;

            float v1 = rychlost1.x;
            float v2 = rychlost2.x;

        }

    }

    public void ButtonPressed()
    {
        UnityEngine.Debug.Log("Pressed");

        startTime = DateTime.Now;

        GameObject govelocity1 = GameObject.Find("rych1");
        InputField text = govelocity1.GetComponent<InputField>();
        if (text != null)
        {
            string stext = text.text;
            rychlost1.x= float.Parse(stext);
        }

        GameObject govelocity2 = GameObject.Find("rych2");
```

```
            text = govelocity2.GetComponent<InputField>();
            if (text != null)
            {
                string stext = text.text;
                rychlost2.x = float.Parse(stext);
            }

        }

        public void ResetButton()
        {
            UnityEngine.Debug.Log("Pressed reset");

            rychlost1 = new Vector2(0, 0);
            rychlost2 = new Vector2(0, 0);
            beforeCrash = true;


            koule1.transform.position = new Vector2(-50, 0);
            koule2.transform.position = new Vector2(10, 0);

            startTime = DateTime.MinValue;
        }
    }
```

Příloha 2: Zdrojový kód k úloze 2 a 3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using System.Linq;

public class GameLogic : MonoBehaviour
{
    //public static GameSystem Instance { get; private set; }
    GameObject koule1, koule2;
    Rigidbody2D component1, component2;
    bool beforeCrash = true ;
    DateTime startTime = DateTime.Now;

    public InputField energie1;
    public InputField energie2;

    Vector2 rychlost1 = new Vector2(0, 0);
    Vector2 rychlost2 = new Vector2(0, 0);

    float hmotnost1 = 5f;
    float hmotnost2 = 5f;

    // Start is called before the first frame update
    void Start()
    {
        koule1 = GameObject.Find("koule1");
        component1 = koule1.GetComponent<Rigidbody2D>();

        koule2 = GameObject.Find("koule2");
        component2 = koule2.GetComponent<Rigidbody2D>();

        ResetButton();

        //this.ButtonPressed();
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        //float.Parse("0.5f");

        UnityEngine.Debug.Log("FixedUpdate at " +
startTime.ToString());
```

```
        #region texty
        var vel1 = GameObject.Find("time");
        Text txt = vel1.GetComponent<Text>();
        double sec = (DateTime.Now - startTime).TotalSeconds;
        if (beforeCrash)
        {
            txt.text = startTime != DateTime.MinValue ?
sec.ToString() : "0";
        }
            txt = GameObject.Find("kin").GetComponent<Text>();
            txt.text = (0.5f * hmotnost1 * (rychlost1.x *
rychlost1.x) + 0.5f * hmotnost2 * rychlost2.x *
rychlost2.x).ToString();

            txt = GameObject.Find("hyb").GetComponent<Text>();
            txt.text = (hmotnost1 * rychlost1.x + hmotnost2 *
rychlost2.x).ToString();

            Vector2 screenPos =
Camera.main.WorldToScreenPoint(component1.transform.position);

GameObject.Find("popisRychlost1").GetComponent<RectTransform>().an
choredPosition = screenPos - new Vector2(100, -70);

GameObject.Find("popisRychlost1").GetComponent<Text>().text =
"Rychlost: " + rychlost1.x.ToString() + "\nHybnost: " + (hmotnost1
* rychlost1.x).ToString()
                + "\nEnergie: " + (0.5 * hmotnost1 * rychlost1.x *
rychlost1.x).ToString();

            screenPos =
Camera.main.WorldToScreenPoint(component2.transform.position);

GameObject.Find("popisRychlost2").GetComponent<RectTransform>().an
choredPosition = screenPos - new Vector2(-100, -70);

GameObject.Find("popisRychlost2").GetComponent<Text>().text =
"Rychlost: " + rychlost2.x.ToString() + "\nHybnost: " + (hmotnost2
* rychlost2.x).ToString()
                + "\nEnergie: " + (0.5 * hmotnost2 * rychlost2.x *
rychlost2.x).ToString();


        #endregion
```

```
        component1.transform.position = new
Vector2(component1.transform.position.x + rychlost1.x *
Time.fixedDeltaTime, component1.transform.position.y + rychlost1.y
* Time.fixedDeltaTime);

        component2.transform.position = new
Vector2(component2.transform.position.x + rychlost2.x *
Time.fixedDeltaTime, 0);

        if (beforeCrash && (koule1.transform.position.x + 3 >
koule2.transform.position.x)) //srazka
        {
            beforeCrash = false;

            float v1 = rychlost1.x;
            float v2 = rychlost2.x;

            float v1new, v2new;
            if
(GameObject.Find("pruzn").GetComponent<Toggle>().isOn)
            {
                v1new = (hmotnost1 * v1 + hmotnost2 * (2 * v2 -
v1)) / (hmotnost1 + hmotnost2);
                v2new = (hmotnost2 * v2 + hmotnost1 * (2 * v1 -
v2)) / (hmotnost1 + hmotnost2);
            }
            else
            {
                v1new = (hmotnost1*rychlost1.x +
hmotnost2*rychlost2.x)/(hmotnost1+hmotnost2);
                UnityEngine.Debug.Log(v1new);
                v2new = v1new;
            }


            rychlost1 = new Vector2(v1new, 0);
            rychlost2 = new Vector2(v2new, 0);
        }
    }

    public void ButtonPressed()
    {
        startTime = DateTime.Now;
        UnityEngine.Debug.Log("Pressed at " +
startTime.ToString());



        GameObject gomass1 = GameObject.Find("mass1");
```

```
        InputField text = gomass1.GetComponent<InputField>();
        if (text != null)
        {
            string stext = text.text;
            hmotnost1 = float.Parse(stext);
        }

        GameObject gomass2 = GameObject.Find("mass2");
        text = gomass2.GetComponent<InputField>();
        if (text != null)
        {
            string stext = text.text;
            hmotnost2 = float.Parse(stext);
        }

        GameObject goenergy1 = GameObject.Find("energie1");
        InputField text2 = goenergy1.GetComponent<InputField>();
        if (text2 != null)
        {
            string stext2 = text2.text;
            rychlost1.x = float.Parse(stext2);
        }

        GameObject goenergy2 = GameObject.Find("energie2");
        text2 = goenergy2.GetComponent<InputField>();
        if (text2 != null)
        {
            string stext2 = text2.text;
            rychlost2.x = float.Parse(stext2);
        }

    }

    public void ResetButton()
    {
        UnityEngine.Debug.Log("Pressed reset at " +
startTime.ToString());

        rychlost1 = new Vector2(0, 0);
        rychlost2 = new Vector2(0, 0);
        beforeCrash = true;


        koule1.transform.position = new Vector2(-30, 0);
        koule2.transform.position = new Vector2(30, 0);

        startTime = DateTime.MinValue;
    }
```

}

Příloha 3: Zadání úlohy jedna a výzkumných otázek respondentům

Dva hmotné body se pohybují ve směru osy x rovnoměrným přímočarým pohybem tak, že v počátečním čase $t = 0$ jsou od sebe vzdáleny o $s = 60$ m a velikosti jejich rychlostí jsou:

c) $v_{s1} = 5\ \mathrm{m \cdot s^{-1}}, v_{s2} = 2\ \mathrm{m \cdot s^{-1}}$;

d) $v_{s1} = 5\ \mathrm{m \cdot s^{-1}}, v_{s2} = -2\ \mathrm{m \cdot s^{-1}}$.

Určete místo a čas, ve kterém se v obou případech potkají.

1. Jsem:
    a. Kluk b. Holka
2. Zadání úlohy jedna mi přišlo srozumitelné.
    a. Ano, velmi b. Spíše ano c. Spíše ne d. Vůbec ne
3. Využití aplikace v Unity mi pomohlo porozumět zadání úlohy jedna.
    a. Ano velmi b. Spíše ano c. Spíše ne d. Vůbec ne
4. Úloha jedna byla v aplikaci přehledná.
    a. Ano velmi b. Spíše ano c. Spíše ne d. Vůbec ne