UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Systém vyhodnocování terapeutických metod pomocí zařízení Kinect

Bc. Jindřich Zerzánek

Diplomová práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Jindřich Zerzánek**
Osobní číslo:	**I11425**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém vyhodnocování terapeutických metod pomocí zařízení Kinect**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V úvodní části práce je nutné provést rešerši systémů využívajících snímání pohybu pro oblast terapie. Další část musí obsahovat analýzu a návrh řešení problematiky pomocí UML diagramů, přehled použitelných technologií. Cílem diplomové práce je navržení a vytvoření programové aplikace pro terapeutické účely využívající možnosti pohybového senzoru Kinect: - Edukativní vzdělávání pro děti předškolního věku a seniory. - Vyhodnocování fyzioterapických postupů, mentálního zdraví apod. - Procvičování na souboru úloh v oblasti pohybu, logiky a paměti. Aplikace se otestuje v praktickém využití, vyhodnotí a výsledky zveřejní. Pro vytvoření aplikace bude využita platforma .NET a programovací jazyk C♯.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. NAGEL, Ch., EVJEN, B., GLYNN, J., WATSON, K., SKINNER, M. C# 2008. Programujeme profesionálně. Computer Press, 2009 2. MICROSOFT. Knihovna MSDN [online]. [cit. 2012-10-08]. Dostupné z: http://msdn.microsoft.com/cs-cz/library/default.aspx 3. MICROSOFT. Develop for Kinect Microsoft Kinect for Windows [online]. [cit. 2012-10-08]. Dostupné z: http://www.microsoft.com/en-us/kinectforwindows/ 4. CZOMPÁL, Zsolt. Ovládání desktopu pomocí senzoru Kinect [online]. Brno, 2012 [cit. 2012-10-08]. Dostupné z: http://www.fit.vutbr.cz/study/DP/DP.php id=11332&y=2011&k=Kinect. Diplomová práce. FIT VUT Brno. Vedoucí práce Pavel Žák. 5. JETENSKÝ, Pavel. Use your desk as mouse-free multitouch interface with MS Kinect [online]. 2011 [cit. 2012-10-08]. Dostupné z: http://jetensky.net/blog/2011/12/18/touchtable/more-140

Vedoucí diplomové práce: **Ing. Zbyněk Kopecký**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2012

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 17. 6. 2013 Bc. Jindřich Zerzánek

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Zbyňkovi Kopeckému za rady, připomínky a čas, který mi věnoval. Dále bych chtěl poděkovat PaedDr. Zdeňce Šándorové, Ph.D. za pomoc výběru úloh na procvičování paměti, pohybu a logiky. Na závěr i svým rodičům a přítelkyni za veškerou podporu, která se mi dostala během celého studia.

Anotace

Tato práce je zaměřena na vytvoření programové aplikace pro terapeutické účely využívající možnosti pohybového senzoru Kinect. Jde tedy o systém, který umožňuje dětem předškolního věku a seniorům procvičovat na souboru úloh, oblasti paměti, pohybu a logiky. První část práce je teoretická, zde je řešena problematika získávání dat a aplikovatelnost sensoru Kinect. Druhá část se zabývá samotnou realizací systému, kde se jedná o popis důležitých části.

Klíčová slova

Kinect, terapie, edukativní vzdělávání, fyzioterapické postupy, pohybový senzor

Title

The assessment system of therapeutic methods by using Kinect

Annotation

This work is focused on creating software application for therapeutic purposes using the motion sensor Kinect. It is therefore a system that allows children of preschool age and seniors practice on a set of tasks in memory, movement and logic. The first part is theoretical there is the issue of data acquisition and applicability of the Kinect sensor. The second part discusses the implementation of the system, where it is a description of the important parts.

Keywords

Kinect, therapy, educational training, physiotherapy practice, motion sensor

Obsah

Seznam zkratek

.NET	Dotnet
C#	C Sharp
ERD	Entity-relationship diagram
ICT	Information and Communication Technologies
WPF	Windows Presentation Foundation
SDK	Software development kit
A/D	Analogově-digitální
MVC	Model-view-controller
PNG	Portable Network Graphics
JPEG	Joint Photographic Experts Group
XAML	Extensible Application Markup Language

Seznam obrázků

1 Úvod

Aplikováním nových informačních technologií do širšího spektra využití se setkáváme denně. Princip spočívá v tom, že se vytvoří nová technologie a hledá se pro ni uplatnění ve všech směrech. Jako názorný příklad může sloužit pohybový senzor Kinect vyvinutý společností Microsoft Corporation, který souvisí s touto prací. Prvotní účel tohoto zařízení spočíval v herním průmyslu a následně se začal rozšiřovat do ostatních odvětví, jako je zdravotnictví, obrana, umění, školství. [1] Téma Systém vyhodnocování terapeutických metod pomocí zařízení Kinect, již napovídá, že se zabývá uplatněním ve fyzioterapii, přesněji se zaměřuje na procvičování oblastí paměti, pohybu a logiky.

Pro dosažení výsledku diplomové práce jsou třeba znalosti z několika předmětů a to z projektování softwarových systému, UML, pokročilých technik programování a databázových systémů. Dalším krokem je propojení pohybového senzoru s počítačem za účelem získávání potřebných dat, která se zpracovávají, jak na ovládání aplikace, tak na snímání částí těla.

Cílem diplomové práce je navržení a vytvoření programové aplikace pro terapeutické účely využívající možnosti pohybového senzoru Kinect. Umožňuje edukativní vzdělávání pro děti předškolního věku a seniory, vyhodnocování fyzioterapických postupů, mentálního zdraví. Procvičování na souboru úloh v oblasti pohybu, logiky a paměti. Následně vyhodnotí výsledky. Pro vytvoření aplikace je využita platforma .NET a programovací jazyk C#.

Práce se dělí na dvě hlavní části, teoretickou a praktickou. První část obsahuje pojmy související s tímto tématem, poté analýzou systémů využívajících snímání pohybu pro oblast terapie, pro upřesnění jejich výhodami a nevýhodami. Dále pak podrobnostmi o samotném zařízení Kinect, tedy o jeho technologii, vývojových nástrojích atd. V druhé části je obsažen návrh systému a jeho implementace, následně i praktické otestování jednotlivých částí systému.

2 Rozbor zadání a cílů

2.1 Rozbor

Vývojem různých desktopových aplikací využívajících pohybový senzor Kinect je již dnes spousty, i když tato technologie je poměrně nová. V České republice jde pořád o zatím málo známou technologii a už vůbec nepoužívanou pro odvětví fyzioterapie. Úkolem této práce je vytvořit systém, který umožňuje procvičit oblasti paměti, pohybu a logiky za pomoci her. Jelikož jde o rozsáhlé oblasti, je systém navržen tak, aby bylo možné vytvářet další moduly bez nutnosti složitějšího zásahu. Například trénování paměti jde mnoha způsoby, v této aplikaci je však ukázka hry na procvičování krátkodobé paměti

2.2 Cíle

- Rešerše systémů využívajících snímání pohybu pro oblast terapie.
- Analýza a návrh řešení problematiky pomocí UML diagramů.
- Navržení a vytvoření programové aplikace pro terapeutické účely využívající možnosti pohybového senzoru Kinect.
- Edukativní vzdělávání pro děti předškolního věku a seniory.
- Vyhodnocování fyzioterapických postupů a mentálního zdraví.
- Procvičování na souboru úloh v oblasti pohybu, logiky a paměti.
- Otestování v praktickém využití a vyhodnocení výsledků.

2.3 Pojmy

2.3.1 Terapie

Jde o léčbu, která má různé formy, může jít o podávání medikamentů, operativní zákrok odstraňující potíže, ozařování, pobyt v klidném prostředí v případě psychických potíží, či pouze vhodná forma rozhovoru. Druh terapie se liší v rámci potíží, které jedinec má. Proces léčby může být zakončen vyléčením, či zlepšením pacienta, případně ukončením léčby bez zlepšení zdravotního stavu. [2]

2.3.2 Paměť

Je to schopnost centrální nervové soustavy uchovávat a používat informace o předchozích zkušenostech. Jedná se o proces vštěpování (kódování), uchovávání (retence) a vybavování (reprodukce) zkušeností. Paměť se dělí jednak podle délky doby uchování zapamatovaného na senzorickou, krátkodobou, střednědobou a dlouhodobou. Dále podle formy ukládání informací na vizuální, akustickou, sémantickou (ukládání významu informace) atd. Další variantou je rozdělení paměti na mechanickou a logickou. [3]

2.3.3 Logika

Myšlenková cesta, která vede k daným závěrům. Jde o vědní disciplínu, zkoumající právě onen způsob vyvozování závěrů, tedy správnost lidského myšlení. [4]

2.3.4 Lidský pohyb

Nejobecnější definice vymezují pohyb jako o změnu vzájemného postavení jednotlivých pohybových částí lidského těla, jako změnu polohy, tedy místní změnu přemístění celého organismu v prostoru. [5]

2.3.5 Edukativní vzdělávání

Jde o takové činnosti lidí, při nichž dochází k učení na straně nějakého subjektu, jemuž je exponován nějakým jiným subjektem přímo nebo zprostředkovaně (textem, technickým zařízením aj) určitý druh informací. [6]

2.3.6 Pohybové senzory

V překladu Motion capture je termín, kterým se označuje proces nahrávání pohybu skutečného objektu a jeho převedení na digitální model. Využívá se zejména v zábavním průmyslu, tj. počítačových hrách, filmu, reklamě, pro medicínské a sportovní účely (analýza pohybu, hodnocení vývoje rehabilitace pacientů s těžkým poškozením nervového nebo motorického ústrojí), v zoologii (analýza pohybu různých zvířat a její vývoj se stářím jednotlivce), dále ve vojenství, trenažérech a také pro vyhodnocování funkčnosti designu různých výrobků (např. interiéru automobilů). [7]

Optické systémy na snímání pohybu

Princip spočívá v tom, že je nahrávaný objem obklopen množstvím speciálních kamer a herci jsou označkováni na všech částech těla pomocí speciálních kuliček, markerů. Tyto markery jsou zaznamenávány, nikoli herec sám. Kamery po kalibraci fungují podobně jako lidské vidění. Optický systém na snímání pohybu má zásadní výhodu v mimořádné přesnosti a flexibilitě, protože na co jde nalepit marker, to jde v zásadě sledovat. Zásadní nevýhodou je, že marker je trasován úspěšně jen tehdy, když ho registrují alespoň dvě kamery v daném momentu. Pokud herec marker tělem zakryje, případně se herci vzájemně zastíní, trajektorie markeru je ztracena. [7]

Markery

Zařízení, která se používají u optických sytému na snímání pohybu. Jde o kuličky, jež odrážejí světlo zpátky do infračervené nebo červené kamery, čímž se zjistí přesná poloha markeru. Tyto kuličky (někdy polokoule) mají různou velikost i tuhost podle toho, co se má zrovna zaznamenávat. Nevýhodou tohoto systému je potřeba speciálního studia s řízeným světlem a s tím související omezený prostor pro snímání. V současné době se začínají užívat vedle zmíněných markerů, které se označují jako pasivní, i aktivní (napájené) markery, jež světlo pouze neodrážejí, ale mohou ho navíc rovněž vysílat. [7]

3 Analýza systémů využívajících snímání pohybu pro oblast terapie

Tato kapitola se zabývá již existujícími systémy, které lze použít pro terapii, jež využívají technologií na snímání pohybu.

3.1 Systém Qualisys

Qualisys má v celosvětovém měřítku vedoucí postavení v poskytování produktů a služeb v oblasti optického záznamu pohybu (optical motion capture, mocap). Měřící systém se skládá z rychloběžných přesných kamer a špičkový software pro sledování pohybu a analýzu dat. Technologie produktu firmy Qualisys je vyvíjena od roku 1989. Zkušenosti kvalifikovaného týmu umožnily vznik unikátní platformy pro optický záznam pohybu splňující lékařské i průmyslové standarty. Společnost má dlouhou tradici v poskytování vysoce kvalitních produktů v oboru optického záznamu pohybu a konečnému zákazníkovi poskytuje produkt s optimálním poměrem cena/výkon. Snahou firmy je poskytovat nadstandardní služby stávajícím zákazníkům v oblasti servisu a poradenství. [8]

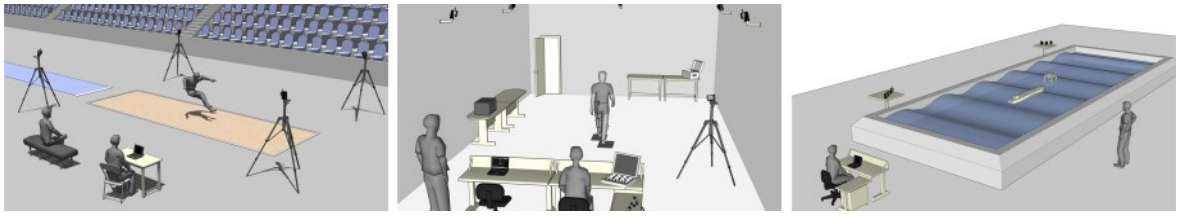
Obrázek 1 – Logo systému Qualisys. Zdroj: [8]

3.1.1 Snímání pohybu

Systém používá vlastní vysokofrekvenční kamery pro přesné sledování pohybu měřeného objektu s využitím pasivních či aktivních markerů. Jedná se o přesnou a velmi robustní technologii, která poskytuje vysoce kvalitní data v reálném čase. Softwarové nástroje umožňují jak snadné výpočty základních kinematických veličin jako dráha, rychlost, zrychlení a úhlové charakteristiky, tak i náročné komplexní kalkulace. [8]

3.1.2 Aplikace systému

Optické snímání pohybu umožňuje měřit pohyb, který by jinak bylo obtížné zaznamenat a vyhodnotit. Optické snímání pohybu je v dnešní době celosvětově přijímáno a využíváno. Využití je velmi široké od výzkumu a klinického využití v lékařství, přes ovlivňování sportovní techniky či četné aplikace k průmyslu až po tvorbu počítačových animací. Například pochopením principů pohybu lidského mohou lékaři a fyzioterapeuti zefektivnit léčbu či zvolit vhodné terapeutické prostředky. Qualisys systém se hojně využívá při průmyslových aplikacích. Lze například měřit a analyzovat komplexní 3D vibrace. Při tvorbě automobilů může být systém využit v rámci vnitřního designu ke zlepšení ergonomie obsluhy vozu a pro hodnocení komfortu a bezpečnosti řidiče. Pro zvýšení interaktivnosti a reálnosti simulovaného prostředí se systém používá při tvorbě počítačových animací a ve virtuální realitě (virtual reality). [8]



Obrázek 2 – Ukázka použití systému Qualisys. Zdroj: [8]

Níže jsou uvedeny příklady využití tohoto sytému v terapii. [8]

Analýza chůze a rehabilitace

Kombinací pokročilých měřících technologií a biomechanického modelování lze nyní objektivně posuzovat lidskou chůzi. Výzkum a vývoj v oblasti analýzy chůze je velmi aktivní obor s rychlým rozvojem poznání. Odborníci z oblasti fyzioterapie, ortopedie a neurologie používají analýzu chůze ke zjištění stavu pacienta a k jeho léčbě a rehabilitaci. [8]

Sport

Optoelektronický záznam pohybu je ideální pro zkoumání pohybu při sportovních výkonech, rehabilitaci, tělovýchově a tréninku. U vrcholových sportovců hrají významnou úlohu faktory jako fyzické limity, optimalizace pohybu. Zkoumání pohybu umožňuje pochopit mechanizmy zranění a jejich prevenci. Může být také použit pro zlepšení sportovní techniky a dosahování lepších výkonů. [8]

Psychologie

Studie pohybu jsou pro psychologický výzkum velmi důležité. Analýza pohybu je výborný způsob jak získat objektivní a kvantitativní data. Qualisys systém poskytuje přesná a spolehlivá data, je flexibilní a přenosný. [8]

Neurologie

Neurologické problémy vzniklé při zranění nebo onemocnění dramaticky ovlivňují schopnost kontrolovat pohyb. Studium pohybových vzorců postižené populace může pomoci jak při diagnóze tak při léčení. [8]

MRI

Qualisys systém může být používán v místnosti s MRI scannerem při použití speciálně upravených kamer. Typická situace je současné měření aktivity mozku a kinematiky během funkčního MRI měření. [8]

3.1.3 Produkty

Qualisys nabízí celou škálu přístrojů, pomůcek a programů pro záznam pohybu a analýzu naměřených dat. Klíčovým prvkem systému jsou Oqus kamera a software Qualisys Track Manager (QTM). Pro pokročilou analýzu získaných dat Qualisys podporuje produkty

jiných výrobců například software Visual3D firmy C-Motion, Inc. Naměřená data mohou být exportována do různých standardních formátů pro použití v produktech dalších výrobců či vlastních zákaznikových programech. Produkty Qualisys jsou vyrobeny tak, aby splňovaly nejvyšší požadavky na kvalitu, jednoduchost, rychlost a přesnost při používání. Systémy produkované firmou Qualisys jsou flexibilní, mobilní a rozšiřitelné a proto se snadno přizpůsobují různým potřebám průmyslu, výzkumu a klinické praxe. [8]

Obrázek 3 – Oqus Qualisys motion capture camera with high-speed video. Zdroj: [8]

3.2 Kinecto Therapy

Jde o nástroj pro fyzickou rehabilitaci, který provádí proces terapie pomocí zábavných, motivačních a odměňovacích her, jež se ovládají pohybem. Kinecto Therapy je vlastně inspirací této diplomové práce, která čerpá jeho myšlenku. Tento systém využívá pro ovládání her pohybový senzor Microsoft Kinect. Hry jsou vytvořeny tak, aby nebyl jejich jediný účel zábava, ale také humanitární přístup. Celý proces spočívá v tom, že využívá kognitivní a motorické činnosti, které jsou nezbytné v 3D celotělovém snímání pohybu. Jelikož jde o jednoduchý systém, u kterého není nutné rutinní sledování a školení, umožňuje přenést rehabilitační proces za hranice nemocnice a pacient jej může použít kdykoliv a kdekoliv. Nevýhoda systému je, že se dá použít pouze jen na některý typ rehabilitace, tedy nedostačuje pro klasickou konvenční léčbu. [9]

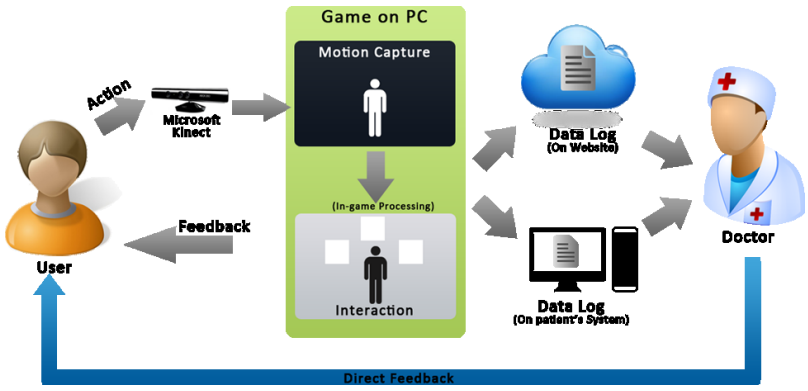
Obrázek 4 – Kinecto Therapy logo. Zdroj: [9]

3.2.1 Produkt

Tento systém obsahuje řadu miniher a každá spadá do žánru Serious Games (vážné hry). Hry v tomto žánru slouží primárně k jinému účelu, než k pouhému pobavení hráče. Jejich účelem může být propagace, vzdělání či procvičování dovedností. Nejedná se o žánr, ale spíše o kategorii her, do kterých je možno řadit hry všech ostatních žánrů, pokud jejich primární účel není zábava. Zábavné prvky přesto vážná hra obsahovat může. [10]

Současné rehabilitační hry postrádají zábavu a poutavé vlastnosti zatímco populární hry postrádají základní složky pro rehabilitaci. Mícháním užitečných prvků obou možností, může vzniknout zábavná a přitom vážná hra. V tomto stylu jsou právě vytvořeny minihry tohoto systému a tak umožňují rehabilitační cviky zábavnou formou, jež využijí fyzioterapeuti a ergoterapeuti u svých pacientů. Minihry jsou zaměřeny na konkrétní problémy u pohybu končetin. Hra se stává živější díky hernímu avataru osoby, která používá systém v reálném čase. Kompletní kostra uživatele je sledována a její pohyb je simulován ve virtuálním prostředí. Takovéto interaktivní prostředí vybízí člověka, aby se více zapojil, motivoval a byl ponořen do rehabilitace. Tyto typy virtuálních prostředí nejsou v konvenční léčbě, ale za to poskytují snadnou a více zábavnou terapii, která může být přizpůsobena zájmům pacientů a jejich fyzických schopností. [9]

Další výhoda tohoto produktu je, že poskytuje zpětnou vazbu k pacientovi na jeho výkon, tedy jak úspěšné bylo dokončení cvičení. Zpětná vazba umožňuje hráči měřit jeho pokrok při dosahování cílů či progresi jeho schopností v průběhu času a to mu dává pocit úspěchu. Při provádění cvičení pacientem jsou různé aspekty o jeho výkonu zaznamenány. Tato data pomáhají při srovnávání jeho současného výkon oproti dřívějším pokusům a následně ho motivuje k lepším výkonům v následujících relacích. Data jsou pravidelně aktualizována na webových stránkách a tak lékař může jednoduše dohlížet na pacienta a zjistit zvláštnosti z jeho statistik na dálku. Toto umožňuje efektivní sledování pacientova pokroku v průběhu času a umožní lékaři rozhodnout se o dalších možnostech terapie. [9]



Obrázek 5 – Schéma architektury Kinecto Therapy. Zdroj: [9]

3.2.2 Minihry

V této časti, jsou popsány některé vybrané minihry, jež jsou součástí systému Kinecto Therapy.

Shoulder exercise

Hra se snaží cvičením zvýšit míru abdukce (upažení) ramene. Pacientův pohyb se promítá na avatarovi, reprezentující herní postavu. Testuje se, zda upaží správně, pokud ano, zvýší se skóre a při splnění daného počtu cviků, dojde k ukončení hry. Zobrazí se výsledky s podrobnými statistikami i počtu špatného upažení. [9]

Obrázek 6 – Ukázka z minihry na trénování abdukce ramene. Zdroj: vlastní

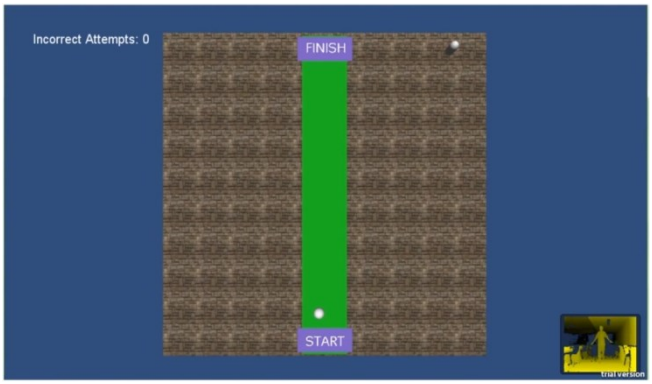
Balloon Burst!

Praskání balónků pomocí ruky, které zvýší stabilitu ramene a prodlužuje pacientův dosah v rovině X-Y. Princip spočívá v tom, že pohyb pacientovy ruky se přenese do hry, po najetí na balón a držením určité doby, dojde k jeho prasknutí. [9]

Obrázek 7 – Ukázka z minihry Balloon Burst. Zdroj: vlastní

Line Control

Cesta následovníka je podání hry na zlepšení rovnováhy a koordinaci při klasické chůzi v přímém směru. Pacientův pohyb je přenesen do hry jako kulička, jež se musí dostat ze startu do cíle, tak že se pohybuje jen ve vytyčeném prostoru. Každé vychýlení je bráno jako chyba a tím se zhorší hodnocení výsledku. [9]



Obrázek 8 – Ukázka z minihry Line Control. Zdroj: vlastní

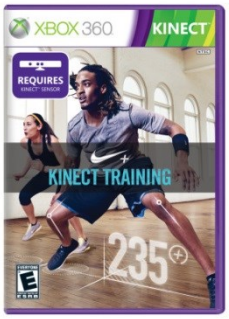
3.3 Microsoft hry

Výrobce pohybového senzoru Kinect, tedy společnost Microsoft Corporation, pro herní konzoly Xbox 360, vydala několik her, které se dají používat pro fyzioterapii. Fyzikální terapie může být dlouhá a obtížná cesta, u které je úspěch těžké vidět skrze každodenní trénink. Kromě toho jde o to, že pacienti navštěvují svého lékaře pouze několikrát v průběhu celého procesu obnovy, který může zahrnovat měsíce až roky. Proto vytvořil edici her Kinect Fitness, pomocí kterých se dá trénovat a zjišťovat pacientův pokrok. Kinect je totiž schopen sledovat tělesný pohyb bez nutnosti aplikování čidel přímo na pacienta. [8] Lze se tedy snadno rehabilitovat z pohodlí domova zábavnou formou.

Zde jsou uvedeny některé tituly z edice Kinect Fitness. [10]

3.3.1 Nike+ Kinect Training

Hra vyvinutá od základu společností Nike s využitím bohatých zkušeností této společnosti na poli atletiky. Umožní provádět osobní trénink v dobře známém prostředí domova a to bez ohledu na fyzickou kondici. Pomocí technologie Kinect vidíme, jak se tělo pohybuje, posoudí fyzickou sílu a ohebnost a zjistí, co je potřeba zlepšit, hra poté vytvoří vlastní cvičební plán ušitý na míru potřebám hráče. [10]



Obrázek 9 – Nike+ Kinect Training for Xbox 360. Zdroj: [9]

3.3.2 Kinect Sports Season Two

Obsahuje několik sportovních disciplín, které zabaví celou rodinu na dlouhé hodiny. Režim Party umožňuje jednotlivým účastníkům vstupovat i vystupovat, kdykoli se jim

zamane. Na výběr jsou následující sporty jako, fotbal, golf, baseball, tenis, šipky a lyžování. [9]

Obrázek 10 – Kinect Sports: Season Two. Zdroj: [9]

3.3.3 Dance Central 3

Taneční hra pro Kinect, která přináší nejúchvatnější zábavu díky více než 40 skladbám, od disco rytmů 70. let až po aktuální taneční pecky. Hra umožní naučit se oblíbené starší taneční hity a zvládnout kroky a pohyby žhavých tanečních novinek. [9]

Obrázek 11 – Dance Central 3 Xbox 360 Game for Kinect. Zdroj: [9]

3.4 Shrnutí

V této části je provedeno shrnutí výše uvedených systémů využívajících snímání pohybu pro oblast terapie a porovnání s výsledným systémem této diplomové práce. Jako první je uveden Qualisys, jeho velkou výhodou je používání optického snímání pohybu a již dlouholetá zkušenost s touto problematikou. Nevýhoda spočívá v komplexnosti a složitosti, je nutné pro použití mít několik kamer, markery, výkonné počítače, software pro analýzu dat, veliký prostor a proškolenou obsluhu. Jako další systém je uvedený Kinecto Therapy, ten na rozdíl oproti předešlému Qualisys, používá novou technologii Microsoft Kinect a vyhne se tedy nevýhodám, které jsou uvedeny. Tento nástroj je vytvořen přímo pro oblast terapie a snaží se pacienty zaujmout zábavnou formou. Nespornou výhodou je, že potřebuje k svému fungování Microsoft Kinect, počítač s operačním systémem Windows 7 či vyšším a veškerý software, tedy minihry běží v internetovém prohlížeči. Bohužel i tento systém není dokonalý, nevýhodou je malý počet vytvořených miniher, bez možnosti dalšího rozšíření a zaměření těchto her jen na pohybové procvičování. Jako posledním systémem k porovnání jsou uvedeny Microsoft hry, které používají pro snímání pohybu

také technologii Microsoft Kinect. Opět jde jako u Kinecto Therapy k zábavnému procvičování pohybu. Na rozdíl od něj jsou tyto hry více sportovní, výběr je rozsáhlejší a grafická úroveň daleko vyšší, protože nejsou spouštěny v internetovém prohlížeči, ale jako samostatné aplikace. Výhoda tedy spočívá v použité technologii a jednoduchosti ovládání. Kromě snímání pohybu je nutná herní konzole Xbox a zobrazovací zařízení, například televize či monitor. Nevýhoda tohoto systému oproti předešlým je nemožnost evidovat výsledky průběhu terapie a tedy žádný zpětný záznam pro terapeuta. Po zhodnocení těchto uvedených systémů se výsledný systém teto práce, snaží vyvarovat jejich nevýhodám, jelikož je postavený na technologii Microsoft Kinect nese s sebou výhody jednoduchého nasazení a používání. Umožňuje jednoduché přidávání nových her, pokrývá širší oblasti terapie a to jak pohyb, logiku tak i paměť. Uchovává pomocí jednoduché databáze veškeré výsledky o průběhu terapie. A je jednoduše přenositelný, protože je postaven na technologii .NET Framework 4.

4 Pohybový senzor Microsoft Kinect for Windows

Microsoft Kinect (Obrázek 12) je zařízení, které umožňuje snímání 3D scény pomocí hloubkového senzoru. Dřívější kódové označení bylo Projekt Natal. Tento senzor byl vytvořen původně k herní konzoli Xbox 360, jako nový způsob ovládat hry pomocí pohybů těla, případně hlasem. Následně poté byla vytvořena i verze pro počítače s operačním systémem Windows, z toho vznikl název Kinect for Windows. Tyto dvě verze se liší minimálně, největším rozdílem je vzdálenost senzoru od snímaného objektu. Verze pro Windows umožňuje snímat již od 40cm, natož verze pro Xbox potřebuje vzdálenost 180cm od objektu. [13]

Obrázek 12 – Pohybový senzor Kinect for Windows. Zdroj: [14]

Účelem Microsoftu bylo mít konkurenční zařízení ke své konzoli, jelikož obdobný způsob ovládání pohybem měly již konzole u ostatních výrobců, jako například Sony u své konzole PlayStation 3 má zařízení PlayStation Move (Obrázek 13). Další konkurenční společností je Nintendo, které má u konzole Wii zařízení Wii Remote Plus (Obrázek 14).

Obrázek 13 – Zařízení PlayStation Move. Zdroj: [15]

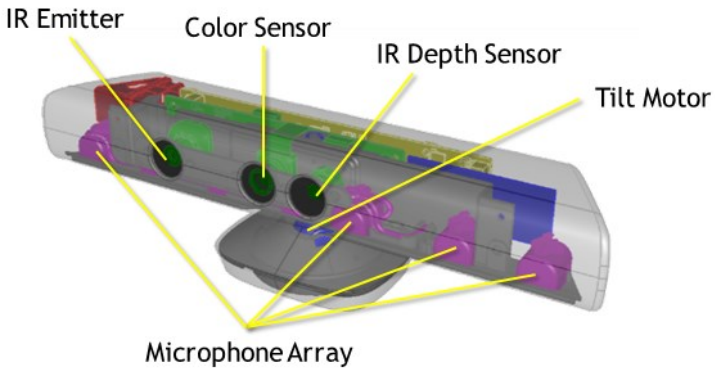
Obrázek 14 – Zařízení Wii Remote Plus. Zdroj: [16]

4.1 Technologie

V této části se nachází popis komponent a principu tohoto pohybového senzoru.

4.1.1 Popis

Zařízení se skládá z mnoha dílů, některé důležité jsou rozebrány níže. Kinect jak bylo vidět na obrázku výše (Obrázek 12) má podstavec a hlavu se senzory, které jsou zde uloženy v horizontální poloze, dále celý obal zařízení je vyroben z plastu.



Obrázek 15 – Komponenty senzoru Kinect for Windows. Zdroj: [14]

Mezi velmi důležité části patří tyto:

- Malý elektromotor, který otáčí a nastavuje hlavu senzoru, umožňuje naklánět pouze ve vertikální ose podle naší potřeby. Úhel náklonu, může být maximálně ±27° od vodorovné polohy. Bohužel by se měl nastavovat co nejméně, není navrhnutý na časté používání. Nastavení úhlu je limitované a můžeme ho tedy nastavit pouze jednou za sekundu. Po připojení zařízení k počítači se provede automaticky naklonění do vhodné polohy, tedy jej není nutné nastavovat manuálně. [14]

- RGB kamera, jež ukládá tři kanálová data v rozlišení 1280×960, případně v jiném rozlišení, které se nastaví. Dále lze pomocí ní vytvářet barevné snímky. [14]

- Infračervený emitor na vyzařování světelných paprsků. [14]

- Infračervený hloubkový senzor na snímání odražených paprsků, které jsou zpracovány a z nich je následně vypočítaná vzdálenost senzoru a objektu. Senzor má nastaven základní rozsah od 800mm do 4000mm, u verze Kinect for Windows je možné přepnutí do blízkého režimu a nastavit tak rozsah od 500mm do 3000mm. [14]

- Čtyři mikrofony, pomocí kterých se zjistí pozice zdroje zvuku a směr zvukové vlny. Jeden mikrofon je na levé straně senzoru, zbylé tři jsou umístěné na pravé. Jsou propojené s 24-bitovým A/D převodníkem a dalšími obvody pro zpracování signálu. Samotný Kinect má zabudovanou funkci na potlačení šumu a zrušení akustických ozvěn. [17]

- Tříosý akcelerometr, který snímá gravitační zrychlení až do dvou G a umožňuje zjistit aktuální orientaci senzoru. [14]

Nevýhoda Kinect for Windows je, že nedokáže fungovat na přímém slunci, dále neumí rozpoznat skleněné předměty, například okna či dveře.

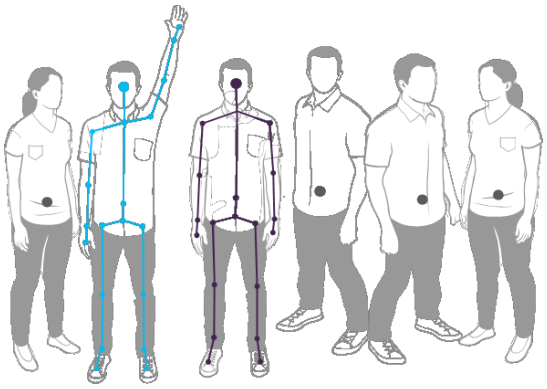
4.1.2 Princip

Zařízení Kinect se řadí do kategorie hloubkových senzorů. To znamená, že detekce okolí je založené na zpracování hloubkové mapy. Hloubková mapa je jednokanálový obraz, ve kterém hodnota každého bodu určuje vzdálenost nejbližšího bodu v prostoru na daných souřadnicích. Kinect využívá pro 3D snímání systém zvaný kódování světla. Infračervený emitor vyšle paprsek světla, který je pomocí krystalů rozdělen na mnoho samostatných paprsků vytvářející konstantní vzor teček promítnutých do scény. Tento vzor je následně zachycen infračerveným senzorem a porovnán s referenčním vzorem uloženým přímo v paměti Kinectu. Díky tomu je možné určit vzdálenost konkrétního objektu od senzoru.

4.2 Vývojové nástroje

Pro vývoj aplikací využívající funkce pohybového senzoru Kinect, je nutné použít knihovnu pomocí, které se přistupuje k datům pořízených tímto zařízením. Existují dvě hlavní knihovny pro práci s technologií Kinect a to OpenNI a Kinect for Windows SDK. Aplikace vytvořená v této práci využívá Kinect for Windows SDK v1.7, což je oficiální knihovna od společnosti Microsoft. Tato knihovna je nekomerční vývojový kit, který obsahuje ovládače kompatibilní s operačním systémem Windows 7 a vyšším. Také kromě ovladačů obsahuje bohaté aplikační programové rozhraní. Poskytuje podporu vývoje softwaru v programovacích jazycích C++, C# a Visual Basic. Podporuje přímý přístup k základním výstupním tokům dat z hloubkového senzoru, RGB kamery i mikrofonům. Nezanedbatelnou součástí je knihovna NUI (Natural User Interface), která využívá základné toky dat a poskytuje pokročilé funkce. Mezi tyto pokročilé funkce patří podpora skeletonizace, transformace souřadnicových systémů, či výpočet roviny podlahy. Skeletonizace je proces extrahovaní aktuálních pozic a směrů základných rysů lidské kostry. Kinect for Windows SDK používá pro skeletonizaci speciální systém, který porovná obrazy lidského těla s tokem dat vznikajícím v hloubkovém senzoru. Tento

způsob umožňuje velmi rychle detekovat objekty tvaru lidského těla a následně z nich generovat sadu rysů. Knihovna podporuje současné sledování maximálně šesti osob, přičemž maximálně dvě z nich dokáže skeletonizovat. [18]



Obrázek 16 – Rozpoznání šesti osob a z toho dvě jsou skeletizované. Zdroj: [14]

Jako další částí vhodnou k tvorbě aplikací využívající senzor Kinect je nástroj Kinect for Windows Developer Toolkit. Tento nástroj obsahuje Kinect Fusion což přináší nové možnosti ovládání pomocí jednoduchých interakčních gest spolu s ukázkovými příklady a API rozhraním, které ušetří programátorům čas. Kromě nástrojů obsahuje tento toolkit ukázky zdrojových kódů, včetně rozhraní umožňující propojit Kinect s programy MATLAB a OpenCV, tedy zefektivní bezdotykový vývoj aplikací. [13]

4.3 Využití

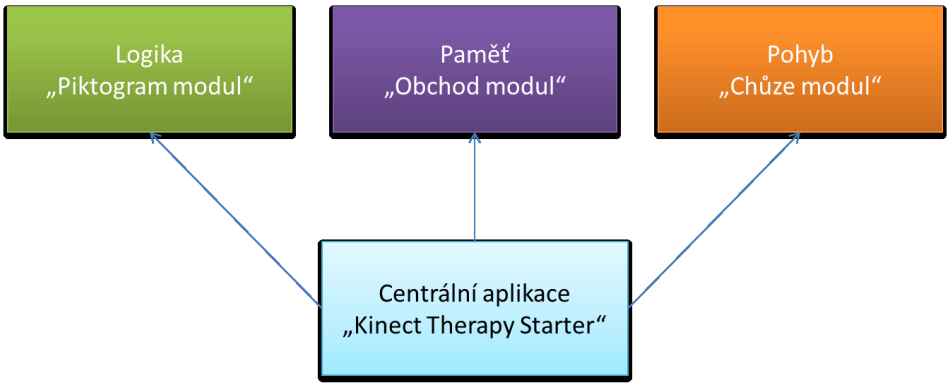
Technologie Kinect byla vyvinuta původně jako nový způsob ovládání her. Ihned po uvedení zdrojových kódů umožňující připojení senzoru k počítači vznikaly projekty, kde všude je možné aplikovat tuto technologii. Našlo se mnoho způsobů využití tohoto zařízení. Například v mobilní robotice, kde se zabývají detekcí překážek, zmapováním místnost ve 3D a rozeznáváním gest lidí a robota. Další zajímavé odvětví je například lékařství, kam patří rehabilitace, která byla zmíněna už v předešlých kapitolách, ale také možnost využití na operačním sále, kdy po celou dobu operativního zákroku musí chirurgovy rukavice zůstat sterilní, aby se zabránilo případné kontaminaci rány. Pokud chce lékař manipulovat s radiologickými snímky bez nutnosti klávesnice a myši, použije jednoduchá gesta a hlasové příkazy. Nejzajímavější aplikací zařízení je v odvětví počítačové grafiky, které používají Kinect pro vytváření 3D modelů podle skutečných předloh v reálném čase. Princip spočívá v tom, že rekonstruovaný objekt postupně otáčíme v zorném poli kamery nebo naopak pohybujeme samotným Kinectem a statický objekt nasnímáme ze všech stran. Vygenerovaný model lze exportovat do běžných formátů a dále s nimi pracovat v jakémkoliv grafickém programu. [17]

5 Návrh systému

V této kapitole jsou prezentovány požadavky funkční a nefunkční. Dále diagramy případů užití, jež zachycují jinou formou požadavků, prezentují co má systém vykonávat a jaké jsou jeho funkce. Dále analytické modely tříd, které jsou potřebné před samotnou implementací. Znázorňují typy objektů a jejich vztahy. Pomocí diagramů tříd lze určit, jaké data systém bude využívat. Podle toho se navrhne datový model, s návrhem tabulek, stanovení relací mezi nimi, určení primárních klíčů. Všechny tyto diagramy vznikly ze zadaných cílů této práce.

5.1 Části systému

Navržený systém se skládá z několika částí. Každý samostatný modul lze spustit bez nutnosti centrální aplikace, ale neuloží do databáze výsledná data. Systém je navržen tak, aby bylo možné jednoduše vytvářet nové a nové moduly. V této práci jsou ukázkově vytvořeny jen tři, pro každou oblast jeden.



Obrázek 17 – Hrubý návrh systému. Zdroj: vlastní

5.1.1 Kinect Therapy Starter

Jde o centrální aplikaci, která uchovává základní záznamy o pacientech, umožňuje zobrazit jejich hodnocení, z něhož lze zjistit, zda dochází ke zlepšení či zhoršení schopností. Dále spouští moduly pro terapii ve vybrané oblasti.

5.1.2 Obchod modul

Tento program obsažený v systému slouží na tréning oblasti paměti, přesněji procvičuje krátkodobou paměť. Princip spočívá v zapamatování si zboží, které je promítnuto na krátkou dobu pacientovi a následně zboží musí vybrat do košíku.

5.1.3 Piktogram modul

Modul určený pro oblast logiky, využívá obrázky zvané piktogramy. Charakterizují pojmy, případně sdělení, jde o malý a srozumitelný nákres. Cílem pacienta je seřazení těchto obrázků do správného logického pořadí. Například může jít o náhodné obrázky s čísly, které se mají správně vzestupně seřadit.

5.1.4 Chůze modul

Aplikace na trénování oblasti pohybu, přesněji stabilizace chůze. Pacient je testován, zda chodí stabilně a nehrozí mu pád. Herní část je tvořena tak, že pacientův pohyb je přenášen na souřadnicové pole a musí dojít na náhodně generované políčko. Samotný pohyb je přenesen jen tehdy, když splní správnou chůzi. Stabilizace je definována několika podmínkami. [19]

1. Osa těla znamená, že ucho a kyčel jsou vertikální (v olovnici).
2. Postavení hlavy, kde uši a oči jsou vyrovnané do horizontály.
3. Pohyb paží je minimálně tak velký, jako dolních končetin, důležitý je také posun paže vzad.
4. Postavení pánve, tedy přední a zadní horní spina iliaca je vyrovnané a vodorovné.
5. Rameno se pohybuje stejným směrem jako paže proti pohybu pánve.
6. Horní končetiny jsou relaxované.
7. Pánev rotuje souhlasně s pohybem dolních končetin proti pohybu ramen.
8. Kolenní kloub stejné nohy je ve středním postavení, není flekován. Postavení v kyčelním a kolenním kloubu je ve fázi opory na jedné noze vyrovnané.

5.2 Technologie pro návrh systému

Veškeré diagramy jsou vytvořeny pomocí aplikace Enterprise Architect od společnosti společnosti Sparx Systems. Jedná se o kompletní nástroj pro systémovou analýzu a návrh, který pokrývá celý životní cyklus vývoje systému, od zadání požadavků přes analýzu stavů, návrh modelů, testování a údržbu, vše s využitím diagramů v UML. Enterprise Architect poskytuje podporu pro týmový vývoj a pro jednotlivé role (analytik, tester, projektový manažer, kontrola kvality, vývojový tým). [20]

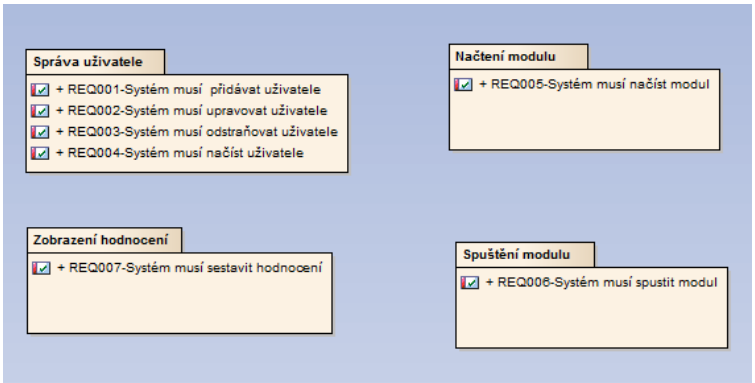
5.3 Požadavky

Specifikují to, co se má v systému implementovat, ale neřeší jakým způsobem. Jejich důležitost je viditelná na pracovních postupech iterace, kde se všechny další postupy přímo odvíjí právě z požadavků. Nedostatečným vypracováním dochází k neúspěchu při nasazování systémů. [20]

5.3.1 Funkční požadavky

Definují to co má systém dělat, jak se má chovat, co má evidovat. [20]

Kinect Therapy Starter



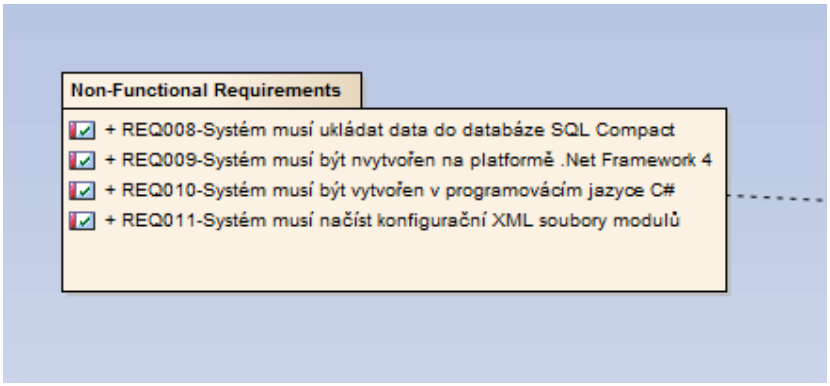
Obrázek 18 – Funkční požadavky aplikace Kinect Therapy Starter. Zdroj: vlastní

V diagramu funkčních požadavků pro centrální aplikaci jsou 4 skupiny, kde každá z nich pokrývá nezbytnou součást systému. Nejpodstatnější z nich je skupina „Správa uživatelů“, která má na starosti kartotéku pacientů. Systém nám musí umožnit přidat nového pacienta/uživatele, případně upravit u něj některé základní informace. Pokud pacient přestane docházet na terapii, měl by být ze systému odstraněn. Dále možnost načtení, která je nutná pro potřebu zobrazení osobních údajů, ukázku výsledků nebo provedení terapie. Jakmile je pacient načten, musí systém umožnit provést terapii pro vybranou oblast. Aby to bylo možné, je potřeba načíst moduly, které jsou v systému a následně spustit vybraný. Jakmile je u pacienta dokončena terapie, musí systém umožnit zobrazit výsledky celkového hodnocení, ze kterého je patrné, zda dochází ke zlepšení, či zhoršení schopností.

5.3.2 Nefunkční požadavky

Nefunkční požadavky definují omezující podmínky systému. Ty většinou vyplívají z prostředí, ve kterém bude systém používán, rychlosti systému, nebo například z hardwarových omezení. [20]

Kinect Therapy Starter



Obrázek 19 – Nefunkční požadavky aplikace Kinect Therapy Starter. Zdroj: vlastní

Mezi nefunkční požadavky této aplikace patří nejpodstatněji uložení dat, že aplikace uchovává data o pacientech a jejich hodnocení v databázi Microsoft Sql Compact 4.0. Dále

systém je vyvíjen v jazyce C# pro platformu .Net Framework 4. Tyto tři aspekty zajistí systému jednoduchou přenositelnost a aplikovatelnost na počítačích s operačním systémem Windows 7 případně Windows 8. Načítání modulu spočívá v tom, že se ze zvolené složky načtou XML soubory, které obsahují informace o daných modulech, jde o jejich názvy, kde jsou umístěny, jaké využívají tabulky atd.

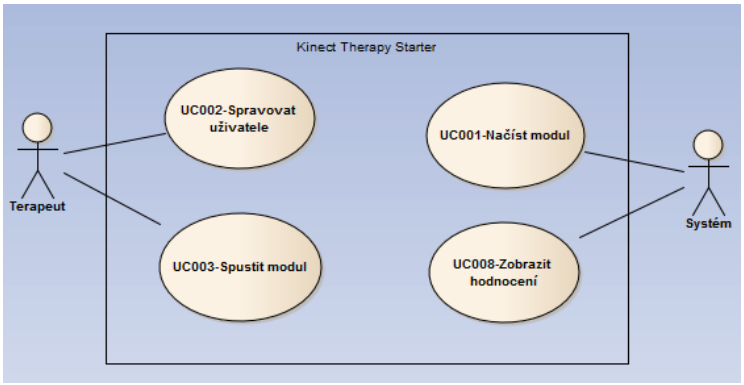
5.4 Model případů užití

Modelování případů užití používá jinou formou zachycení požadavků na systém. Je postaven na aktivitách, jakou jsou nalezení hranic systému, vyhledání účastníků, nalezení případu užití, specifikaci případu užití a tvorbě scénářů. Tyto aktivity jsou rozhodující pro nalezení komponent systému. [21]

5.4.1 Aktéři

Aktéři neboli účastníci nejsou vnímány jako konkrétní osoby, ale z pohledu systému jsou to role, které dané elementy v systému hrají. V systému, jak v centrální aplikaci, tak i v modulech se nachází tři aktéři a to „Systém“, který má na starost logiku aplikací. Dále „Terapeut“ jenž nastavuje moduly, spravuje uživatele a projíždí hodnocení. Posledním účastníkem je „Uživatel“ tedy představitel pacienta, jež používá moduly.

5.4.2 Kinect Therapy Starter



Obrázek 20 – Model případů užití pro aplikaci Kinect Therapy Starter. Zdroj: vlastní

Tento diagram obsahuje dva aktéry. Jelikož se jedná o centrální aplikaci, jejíž význam je popsán výše v kapitole Části systému, je tedy jedním z aktérů terapeut. Jeho náplní je přidávat, odebírat, upravovat a načítat uživatele/pacienta a také pro něj spouštět modul pro vybranou oblast terapie. Další aktér představuje samotný systém, který má za úkol načíst moduly a vytvářet hodnocení uživatelů. Níže jsou rozebrány jednotlivé případy užití.

UC002-Spravovat uživatele

Zajišťuje kompletní správu nad uživateli, ať už jde o vytváření, mazání, odebírání, upravování a načtení. Scénář může vypadat následovně, terapeut zobrazí systémem načtený seznam uživatelů/pacientů, vybere si konkrétního uživatele. Systém načte jeho základní informace, například jméno, příjmení, pohlaví, atd. Nyní terapeut může provést operace jako je odstranit uživatele, pozměnit jeho údaje a uložit jej. Případně vytvořit

nového uživatele. Systém provede pokyn, poté obeznámí terapeuta o výsledném stavu operace informativní hláškou.

UC001-Načíst modul

Systém načítá všechny korektní moduly z konkrétní složky. Jde o konfigurační XML soubory jednotlivých modulů, obsahující údaje o nich. Jde například o název, cestě k aplikaci modulu, tabulku s hodnocením, popis a do jaké oblasti terapie patří. Načítáním vytváří seznam použitelných modulů, který je následně dále využíván dalším částmi systému.

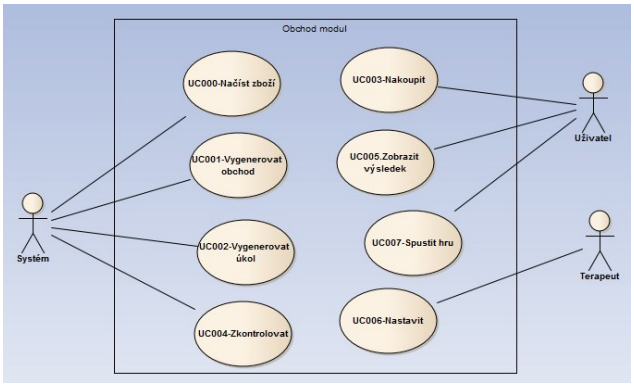
UC003-Spustit modul

Terapeut má načteného konkrétního uživatele a chce pro něj provést terapii. Zobrazí si systémem načtené moduly, vybere si oblast pro trénování a konkrétní modul. Systém spouští aplikaci, pokud nastane chyba, oznámí ji terapeutovi, jinak vybranou aplikaci spustí s parametrem obsahující identifikátor uživatele.

UC008-Zobrazit hodnocení

Pacient dokončil terapii, byl mu zobrazen aktuální výsledek, terapeutovi bohužel jeden nestačí a zajímá ho celkový průběh terapie, proto si u něj v centrální aplikaci zobrazí hodnocení. Systém prochází seznam načtených modulů a vytváří z jejich údajů výstupní výsledek obsahující stavy v jednotlivých modulech. Terapeutovi se objeví okno obsahující prázdný graf a seznam výsledků rozdělených po jednotlivých modulech, s možností zobrazení na grafu, ze kterého je patrné, jak si pacient stojí.

5.4.3 Obchod modul



Obrázek 21 – Model případů užití pro aplikaci Obchod modul. Zdroj: vlastní

Model zobrazující případy užití aplikace modulu Obchod, má obsaženy tři aktéry, opět nejdůležitějším je systém, který řídí celou logiku systému. Má za cíl generovat úkoly, které bude muset uživatel plnit a kontrolovat výsledky. Ještě předtím ale musí načíst data, která mu nastaví terapeut a z nich vygenerovat obchod neboli skupiny zboží. Jako druhý aktér je terapeut, který po spuštění modulu nastavuje cestu k obrázkům, ze kterých se stává zboží.

Posledním účastníkem je uživatel, jenž bude moci spustit hru, nakupovat zboží a i zobrazit výsledek, který vyhodnotí systém.

UC000-Načíst zboží

Systém načte ze složky zadané terapeutem všechny obrázky formátu jpg, ze kterých vytvoří jednotlivé zboží. Z aktuálně zpracovaného obrázku vytvoří název zboží a skupinu zboží z názvu, dále uloží zdrojovou cestu, poté jej jako zboží uloží do seznamu.

UC001-Vygenerovat obchod

Systém vygeneruje obchod, který je tvořen skupinami zboží, pro představu může jít o regál. Každé zboží patří do nějaké skupiny a několik skupin tvoří obchod. Obchod je generován dle nastavení terapeuta, nemusí obsahovat všechny načtená zboží, protože volí, jaké skupiny budou použity.

UC002-Vygenerovat úkol

Systém generuje úkol dle terapeutova nastavení pro uživatele, jakmile je připraven. Vygeneruje seznam zboží, co uživatel bude nakupovat. Každý úkol má jistou obtížnost, ta je tvořena počtem skupin, ze kterého zboží je a počtem zboží, které si má zapamatovat.

UC004-Zkontrolovat

Systém zkontroluje seznam nakoupeného zboží v košíku a zadaného zboží k zapamatování a vyhodnotí výsledek.

UC003-Nakoupit

Uživatel vybírá zboží, které chce koupit a přidává ho do košíku, je limitován počtem zboží, co může koupit, záleží tedy na zadání úkolu.

UC005.Zobrazit výsledek

Uživatel nechal zkontrolovat úkol pomocí sytému a následně mu je zobrazen celkový výsledek úkolu. Ten je také uložen do databáze.

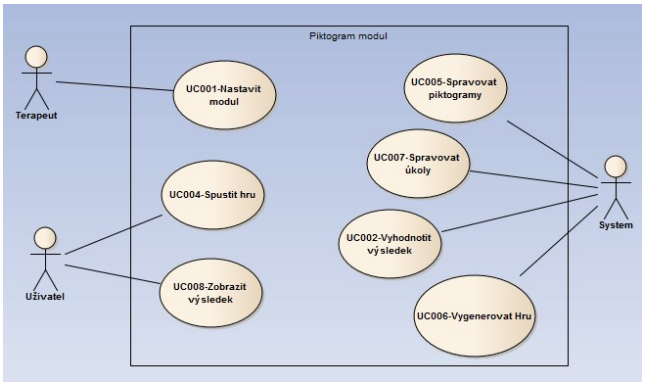
UC007-Spustit hru

Uživatel spustí hru a systém vygeneruje úkol dle nastavení. Promítne zboží na obrazovce v intervalu, jaký mu nastavil terapeut a následně vygeneruje obchod se skupinou zboží, které bude uživatel procházet.

UC006-Nastavit

Terapeut nastaví zdrojovou složku s obrázky, ze kterých systém vygeneruje zboží a skupiny, jež jsou následně terapeutovi zobrazeny. Terapeut si vybere skupiny zboží, které chce nechat jako úkolové uživateli, nastaví čas, jak dlouho má mít na zapamatování a počet zboží.

5.4.4 Piktogram modul



Obrázek 22 – Model případů užití pro aplikaci Piktogram modul. Zdroj: vlastní

Jako další use case model je pro aplikace piktogram modul, kde se vyskytují tři aktéři. Nejvyužívanější systém, který má na starost generování úkolů, správy piktogramů a úkolů. Dále je zde terapeut, jenž má za cíl nastavit modul pro uživatele. A samotný uživatel, který spustí hru a zobrazí si konečný výsledek.

UC001-Nastavit modul

Terapeut nastavuje aplikaci pro uživatele jako například čas na úkol a počet úkolů ke generování.

UC004-Spustit hru

Jakmile je uživatel připraven, spustí hru. Systém vygeneruje uživateli dle nastavení terapeuta hru, která je tvořena úkoly. Celková hra je omezena časem a po uplynutí časového termínu dojde ke zkontrolování.

UC002-Vyhodnotit výsledek

Nastal konec hry, buď uplynul čas, nebo uživatel dohrál. Po ukončení hry systém vyhodnotí správnost výsledku hry, prochází jednotlivé zadané úkoly a porovnává s daty, která zadal uživatel během tohoto procesu a vše ukládá do hodnocení.

UC005-Spravovat piktogramy

Kompletní správa nad piktogramy, jde o přidávání, úpravu, či mazání. Aby mohl systém generovat hru, je nutné mít vytvořené úkoly, které jsou tvořeny piktogramy. Ty spravuje terapeut. Piktogram se skládá z obrázku, názvu a identifikátoru. Pro úpravu či odebrání je nutné zobrazit seznam piktogramů a zvolit. Tyto operace vykonává systém a poté obeznámí terapeuta s výsledkem.

UC007-Spravovat úkoly

Terapeut vytváří nové úkoly, případně je upravuje a odebírá. Každý úkol je tvořen názvem a jednotlivými piktogramy v daném pořadí. Terapeut při vytváření nového úkolu zadá

název a počet políček k úkolu, poté si vygeneruje daný počet políček, do kterých postupně zadává piktogramy ze seznamu. Pro úpravu a odebrání je nutné zvolit ze seznamu úkol.

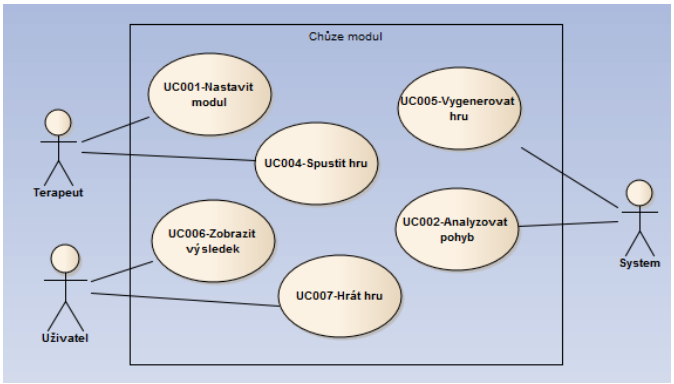
UC006-Vygenerovat Hru

Systém vygeneruje hru, která je složená z několika úkolů, které uživatel poté plní. Generování probíhá na základě terapeutova nastavení (počtu úkolů a času na splnění úkolu). Ve hře nesmí být stejné úkoly.

UC008-Zobrazit výsledek

Systém provedl kontrolu a výsledky uložil do hodnocení. Po dokončení kontroly jsou zobrazeny uživateli výsledky a zároveň uloženy do databáze k ostatním hodnocením.

5.4.5 Chůze modul



Obrázek 23 – Model případů užití pro aplikaci Chůze modul. Zdroj: vlastní

Poslední model případů užití je aplikace chůze modul, která slouží pro kontrolu stability chůze. Obsahuje tři aktéry jako aplikace výše uvedené. Systém má na starost logiku, jedná se o generování hry pro uživatele a vyhodnocení jejího výsledku. Dále nejpodstatnějším cílem systému je analyzovat pohyb, který zpracovává data získaná prostřednictvím zařízení Kinect. Terapeut má za cíl nastavit aplikaci a spustit hru. Uživatel zde hraje hru a zobrazí si výsledky.

UC001-Nastavit modul

Než dojde ke spuštění hry, tak terapeut nastaví aplikaci nastavením počtu úkolu, času na úkol, šířky a výšky herního pole. Dále nastavuje toleranci a počet desetinných míst, se kterými pracuje zpracovávání pohybu.

UC004-Spustit hru

Terapeut spustí uživateli hru, jakmile je obeznámen s pokyny. Systém zahájí odpočet a vygeneruje úkol, po splnění úkolu, generuje nový, vše dle zadaného nastavení.

UC006-Zobrazit výsledek

Jakmile uživatel dokončí hru tím, že splní skóre, nebo uplyne doba na provedení úkolů, systém zobrazí okno s hodnocením, které je tvořeno již během hraní. Po zobrazení dojde k uložení hodnocení do databáze.

UC007-Hrát hru

Terapeut spustil uživateli hru, ve které jde o to, aby plnil podmínky stabilní chůze, jakmile je splní, přenese se pohyb do herního pole jako přesun z políčka na políčko. Pokud se uživatel dostane na políčko červené, získá bod a tím splnil jeden úkol. Každý úspěch je zaznamenán do hodnocení.

UC005-Vygenerovat hru

Systém generuje hru dle nastavení terapeuta, jde o počet políček (rozloha plochy), počet úkolu. Hra generuje nový úkol, jakmile uživatel splní předešlý. Úkol se smí opakovat, ale nesmí být na stejných souřadnicích, jako je uživatel.

UC002-Analyzovat pohyb

Systém zpracovává data získaná zařízením Kinect a vytváří tak analýzu, jež obsahuje výsledky o splnění podmínek pro stabilizace chůze. Pokud je stabilní chůze splněna stoprocentně, uloží jej do seznamu, ze kterého je pak daný počet promítnut do hodnocení hry. Aby se pohyb přenesl do hry, je nutné, aby uživatel splnil určité procento podmínek. Kontrola podmínek probíhá s tolerancí a na počet desetinných míst, které si předem nastavil terapeut v nastavení aplikace.

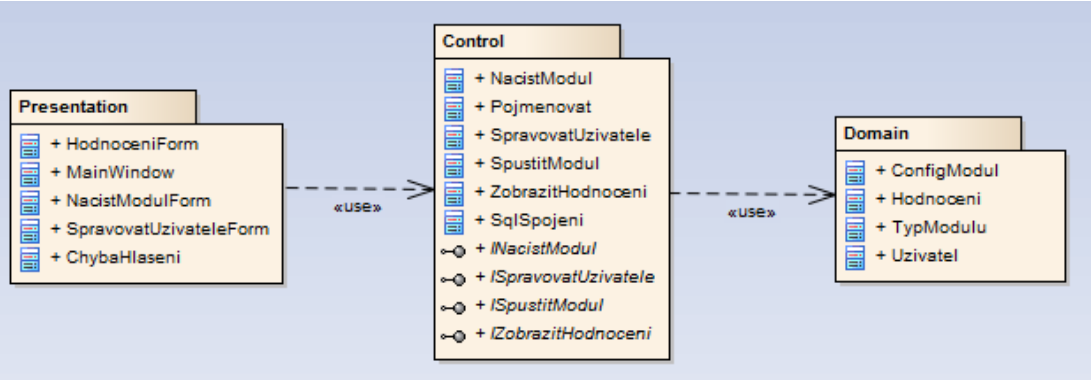
5.5 Analytický model tříd

Jedná se o jeden z prvních kroků pro implementaci aplikace. Diagram zachycuje pravidla modelovaného systému, přesněji znázorňuje typy objektů a jejich vzájemné vztahy. Jednotlivé aplikace systému jsou navrženy tak, aby dodržovaly softwarovou architekturu model-view-controler (MVC). Rozděluje části aplikace do tří nezávislých komponent, jde o datový model, uživatelské rozhraní a řídící logiku, toto zajišťuje že modifikace některé z nich má minimální vliv na ostatní. V této práci je architektura reprezentována vrstvami.

- Presentation − reprezentuje v architektuře část view, obsahuje třídy aplikací stereotypu boundary.
- Domain−reprezentuje v architektuře část model, obsahuje třídy aplikací stereotypu entity.
- Control − reprezentuje v architektuře část controler, obsahuje třídy aplikací stereotypu control.

Jednotlivé třídy aplikací systému jsou rozděleny na entity, boundary, control. Entity třídy představují klíčové abstrakce systému, koncepty a objekty (data), které systém zpracovává. Zobrazují také datový model systému. Objekty těchto tříd se často ukládají do databáze. Boundary třídy reprezentují rozhraní mezi systémem a jeho okolím. Jde nejčastěji o uživatelské rozhraní, také zde patří systémové rozhraní k jiným systémům. Posledním typem tříd je control, který koordinuje správnou funkcionalitu daného případu užití. Obsahuje aplikační logiku, definuje její transakce a jeho účel spočívá v oddělování logiky uživatelského rozhraní od logiky dat. Pro každý případ užití se vytváří jeden controler.

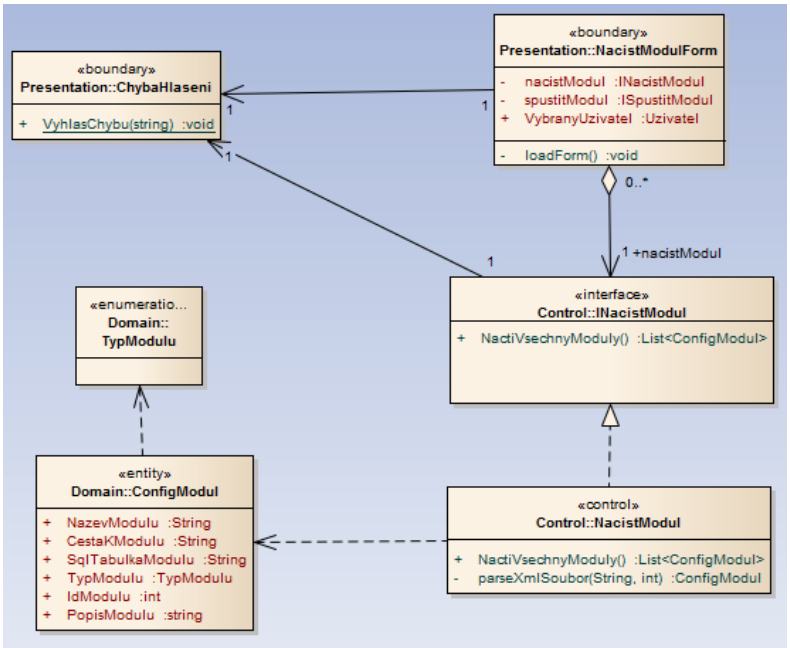
5.5.1 Kinect Therapy Starter



Obrázek 24 – MVC architektura aplikace Kinect Therapy Starter. Zdroj: vlastní

Třídy aplikace Kinect Therapy Starter jsou rozděleny do tří vrstev, aby byla dodržena architektura MVC. Domain obsahuje základní entity, nejdůležitější z nich je `Uzivatel`, obsahuje atributy typické pro charakterizování uživatele (pacienta). Druhým je `ConfigModul`, již z názvu je patrné, že obsahuje atributy pracující s moduly systému. Ve vrstvě Presentation se nacházejí formuláře, se kterými pracuje uživatel systému, v tomto případě terapeut. Nejdůležitějším je zde `MainWindow`, jež je hlavní okno, ze kterého se přistupuje k ostatním formulářům. Poslední vrstva Control zaštiťuje řídící třídy a jejich rozhraní se kterými aplikace pracuje. Jde o samotnou logiku systému, nejzajímavější je zde `SpravovatUzivatele`, jež umožňuje manipulaci s uživatelem, načíst, vytvořit atd. A jako druhý zajímavý je `NacistModul`, který řeší vytváření tříd `ConfigModul` z konfiguračních XML souborů. Níže jsou zobrazeny jednotlivé diagramy tříd některých případů užití.

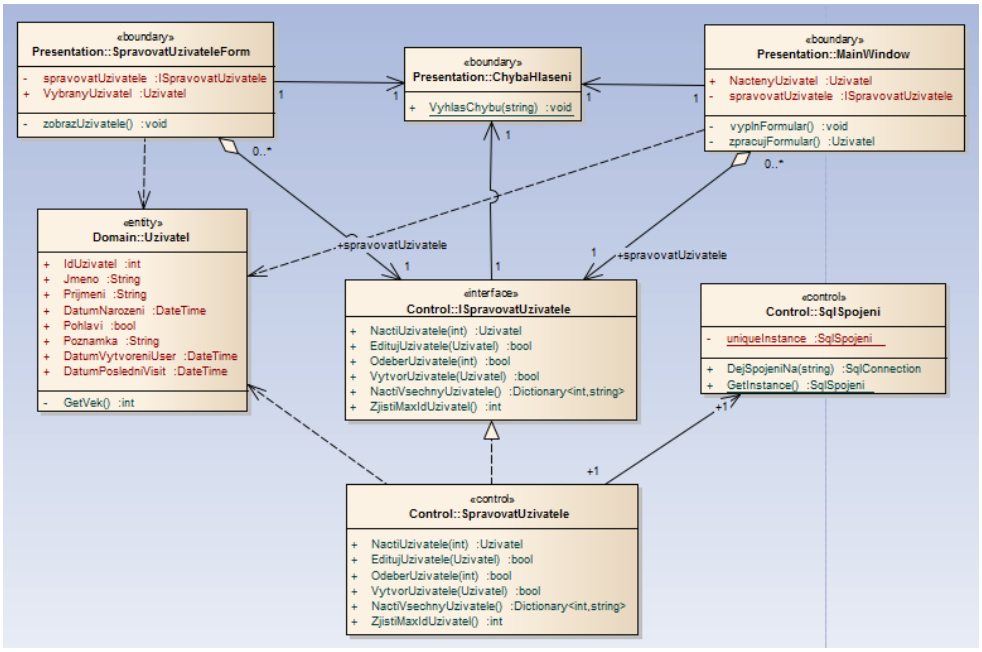
UC001-Načíst modul



Obrázek 25 – Model tříd případu užití UC001-Načíst modul. Zdroj: vlastní

Diagram tříd realizující případ užití UC001-Načíst modul obsahuje následující třídy. Základem je `ConfigModul`, který má atributy potřebné k definování modulu pro terapii přidaného do systému. Jde například o název, popis, ale nejpodstatnější je cesta k aplikaci a tabulka s hodnocením. Využívá také enumerační typ `TypModulu`, který obsahuje tři možnosti, jde o oblasti pro kterou je modul určen, může jít o `Logika`, `Pamet` a `Pohyb`. Nad `ConfigModul` operuje a řídí logiku třída `NacistModul`, jež je implementací rozhraní `INacistModul`. Obsahuje dvě metody, první je veřejná metoda `NactiVsechnyModuly`, která má za úkol vytvořit seznam `ConfigModulů`. K tomu využívá privátní metodu `parseXmlSoubor`, která zpracovává jednotlivé XML soubory a tvoří z nich moduly. Z uživatelského pohledu není tato část vidět, ale výsledek je využit ve formuláři `NacistModulForm`, který obsahuje tlačítka reprezentující jednotlivé moduly ze seznamu, které nám umožní spustit vybraný modul.

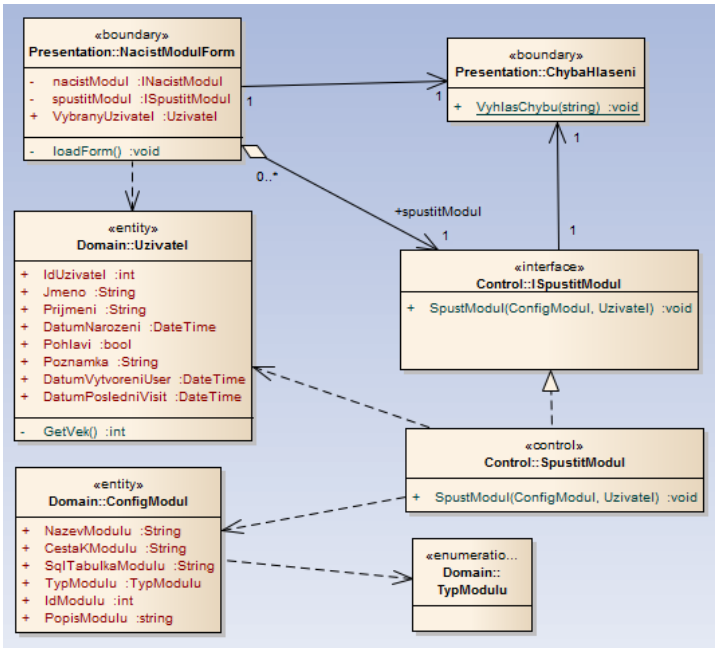
UC002-Spravovat uživatele



Obrázek 26 – Model tříd případu užití UC002-Spravovat uživatele. Zdroj: vlastní

Analytický model pro případ užití UC002-Spravovat uživatele, je složitější než pro načítání modulů. Zde je základním kamenem třída `Uzivatel`, obsahující informace o pacientovi, jde třeba o jméno, datum narození, či pohlaví. Nad touto entitou pracuje control třída `SpravovatUzivatele` jež je implementací rozhraní `ISpravovatUzivatele`. Obsahuje základní operace s uživatelem, Jde o metody `VytvorUzivatele`, `EditujUzivatele`, `OdeberUzivatele`, `NactiUzivatele`, `ZjistiMaxIdUzivatele`, `NactiVsechnyUzivatele`, z názvu je již patrné co která operace dělá. Všechny tyto metody pracují s databází pomocí statické třídy `SqlSpojeni`, která reprezentuje návrhový vzoru Singleton. Pokud dojde k nějakému problému, zobrazí se hláška vyvolaná třídou `ChybaHlaseni`. Tuto logiku používají dva formuláře. První je `MainWindow`, pomocí něhož se vydávají pokyny k operacím a zobrazuje informace o aktuálně načteném uživateli. Druhým je `SpravovatUzivateleForm`, který obsahuje seznam všech uživatelů a umožní tak terapeutovi si vybrat a načíst konkrétního.

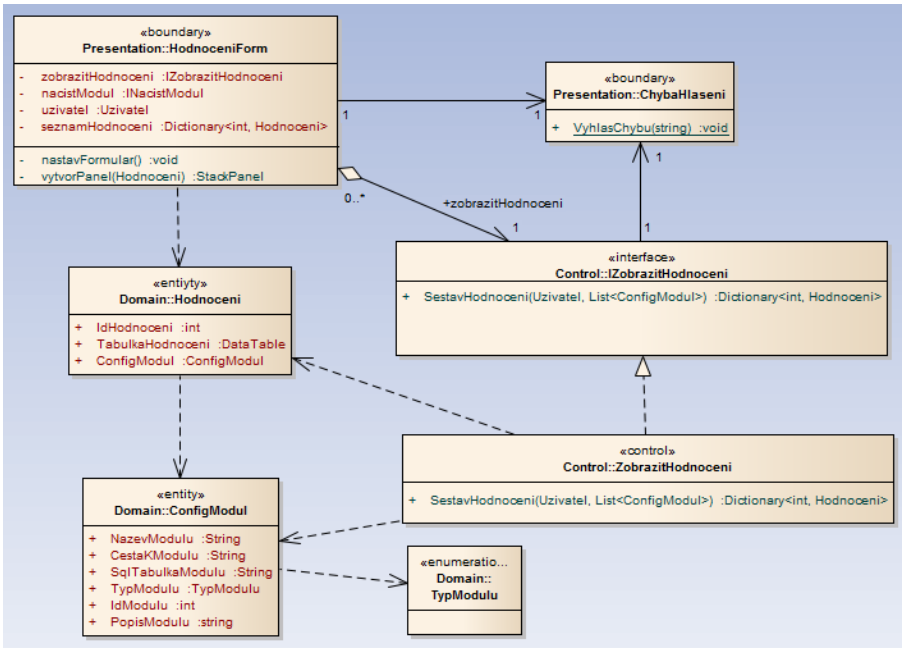
UC003-Spustit modul



Obrázek 27 – Model tříd případu užití UC003-Spustit modul. Zdroj: vlastní

Třídy pro případ užití UC003-Spustit modul, jsou následující. Zde tvoří základ dvě entity a to `Uzivatel` a `ConfigModul`, každá z nich byla rozebrána už výše. Tyto třídy využívá control `SpustitModul`, který je implementací rozhraní `ISpustitModul`, obsahuje metodu `SpustModul`, ta má dva parametry a to typu `Uzivatel` a `ConfigModul`.
Z názvu je patrné, že provede operaci spuštění zvoleného modulu pro dané uživatele. Provede spuštění aplikace uvedené na cestě v modulu s argumentem, který je identifikátor uživatele, je potřeba kvůli uložení hodnocení do databáze po skončení terapie. Při chybě vyhlásí chybovou hlášku třída `ChybaHlaseni`. Pokyn pro spuštění vydává uživatel pomocí formuláře `NacistForum`, jež obsahuje tlačítka reprezentující jednotlivé moduly, načtené pomocí `NacistModul`.

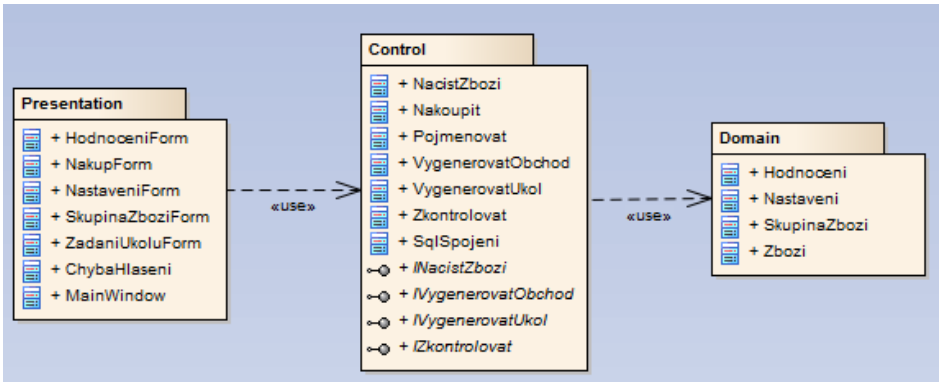
UC008-Zobrazit hodnocení



Obrázek 28 – Model tříd případu užití UC008-Zobrazit hodnocení. Zdroj: vlastní

Poslední model tříd pro případ užití UC008-Zobrazit hodnocení obsahuje následující třídy. Entita Hodnoceni, jež má atribut identifikátor, tabulku s hodnocením a `ConfigModul` charakterizující modul, pro které je hodnocení vytvořeno. Seznam s hodnocením je výstupem metody `SestavHodnoceni` controlu `ZobrazitHodnoceni`, jež je implementací rozhraní `IZobrazitModul`. Tato metoda má za úkol projít seznam s načtenými moduly, přesněji jejich tabulky v databázi a vytvořit z nich tabulky hodnocení jen pro zadaného uživatele. Pokud by nastal problém, zobrazí se hláška pomocí třídy `ChybaHlaseni`. Ale pokud nenastane, tak terapeutovi se zobrazí formulář `HodnoceniForm`, který obsahuje graf a seznam jednotlivých modulů s posledními dvěma výsledky hodnoceních, které vybraný pacient získal. Tento formulář se tvoří dynamicky pomocí metody `vytvorPanel` s parametrem typu Hodnoceni. Vytvoří tedy panel s popiskem názvu modulu, tabulkou hodnocení a tlačítkem, které po kliknutí zobrazí na grafu hodnoty ze všech dosažených hodnocení pro daný modul.

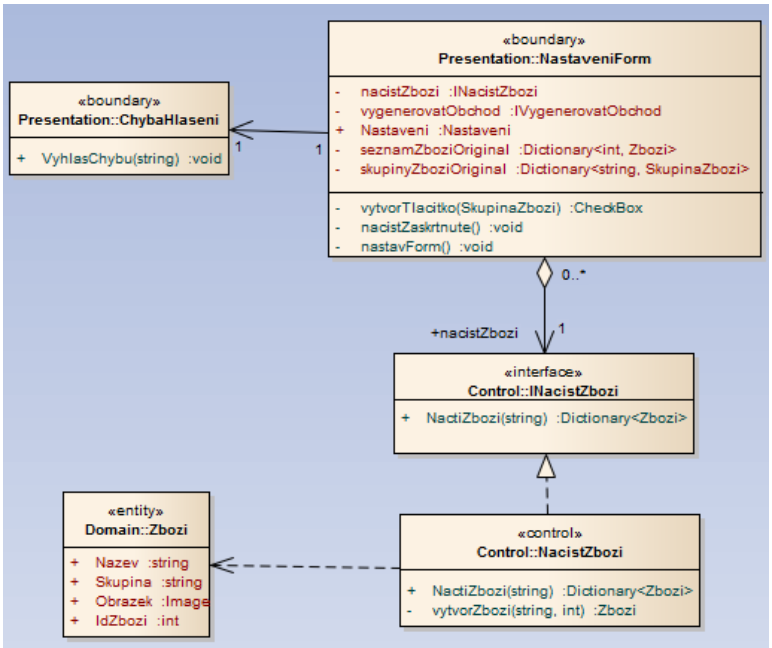
5.5.2 Obchod modul



Obrázek 29 – MVC architektura aplikace Obchod modul. Zdroj: vlastní

Rozdělení tříd aplikace pro trénování krátkodobé paměti Obchod modul. Jde o tři vrstvy, aby byla dodržena architektura MVC. Domain tvoří základní entity, nejdůležitější z nich je `Zbozi`, obsahuje atributy typické pro charakterizování jakéhokoliv zboží. Druhým je `SkupinaZbozi`, z názvu je zřejmé, že seskupuje určitý druh zboží. Ve vrstvě Presentation se nacházejí formuláře, se kterými pracuje jak terapeut, tak i uživatel. Nejdůležitějším je zde `MainWindow`, jež je hlavní okno, ze kterého se přistupuje k ostatním formulářům a i samotná hra zde probíhá. Poslední vrstva Control zaštiťuje řídící třídy a jejich rozhraní se kterými aplikace pracuje. Jde o samotnou logiku systému, nejzajímavější je zde `NacistZbozi`, jež vytváří jednotlivé kusy zboží z jakýchkoliv obrázků. Druhým zajímavým je `VygenerovatObchod`, který vytváří samotný obchod z vybraných skupin zboží. Níže jsou zobrazeny jednotlivé diagramy tříd některých případů užití.

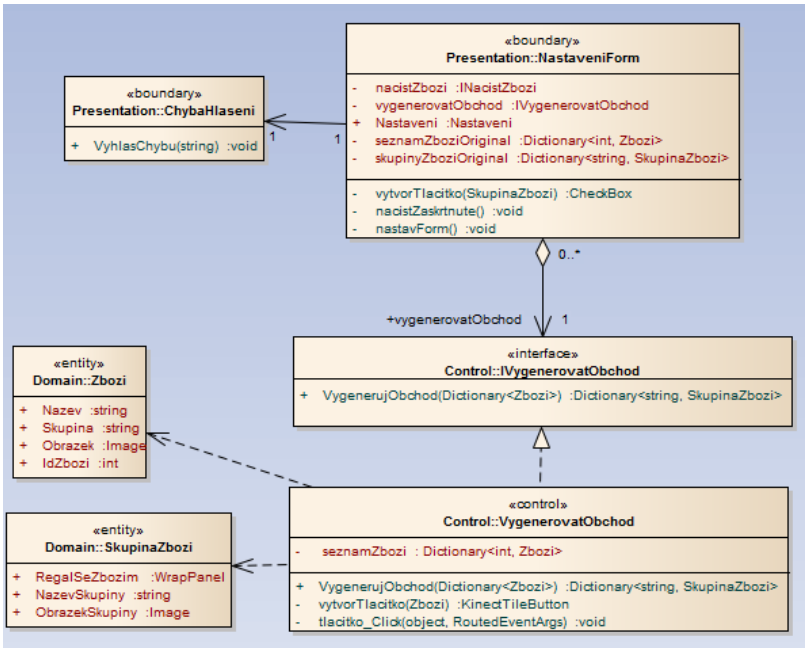
UC000-Načíst zboží



Obrázek 30 – Model tříd případu užití UC000-Načíst zboží. Zdroj: vlastní

Diagram tříd pro případ užití UC000-Načíst zboží obsahuje několik tříd, je postaven na entitě `Zbozi`, která charakterizuji jakékoliv zboží. Mezi její atributy patří identifikátor, název, do jaké skupiny patří a prvotně nejdůležitější je samotný obrázek. Jelikož tento modul je postaven tak, aby nepotřeboval ke své činnosti databázi, vytváří jednotlivé zboží pro hru z libovolných obrázků, které navolí terapeut a jsou typu jpg. Tuto operaci provádí control `NacistZbozi` jež je implementací rozhraní `INacistZbozi`. Má metodu `NactiZbozi` s parametrem zdrojové složky odkud se mají obrázky brát, pokud název obrázku je ve tvaru „text1_text2.jpg“, bere text1 jako název zboží a text2 jako skupiny do které zboží patří, pokud máme „Meloun_Ovoce.jpg“, bude název zboží Meloun a skupina bude Ovoce. Tuto logiku ovládá terapeut prostřednictvím formuláře `NastaveniForm`, které pokud nastane, jakákoliv chyba upozorní chybovou hláškou prostřednictvím třídy `ChybaHlaseni`.

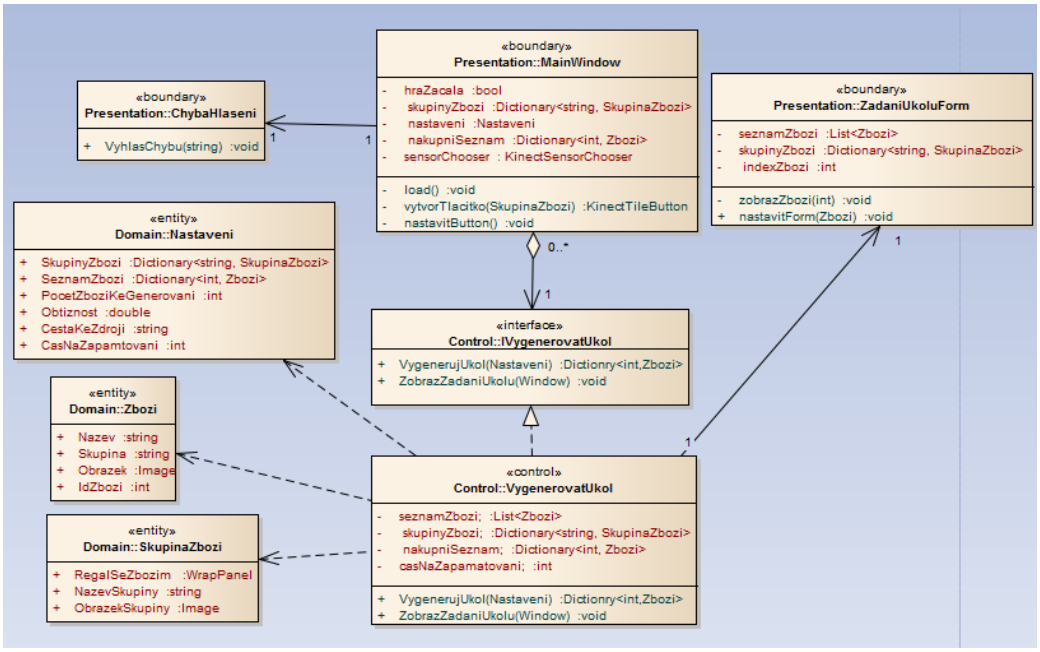
UC001-Vygenerovat obchod



Obrázek 31 – Model tříd případu užití UC001-Vygenerovat obchod. Zdroj: vlastní

Základním kamenem analytického modelu pro případ užití UC001-Vygenerovat obchod jsou třídy `Zbozi` a `SkupinaZbozi`, jedná se o stereotyp entity. Zboží již bylo rozebráno výše, `SkupinaZbozi`, seskupuje zboží stejného typu, pro představu může jít o skupinu Ovoce, do které patří jablko, meloun, banán, atd. Atributem je název skupiny a obrázek charakterizující skupinu, dále regál, ten je tvořen jako panel poskládaný z jednotlivých zboží (jedná se o ovládací prvek tlačítko s obrázkem a popiskem) vedle sebe, pro představu může posloužit opět obchod, jenž má regál, případně pult s ovocem. Nad těmito entitami, pracuje control třída `VygenerovatObchod`, jež je implementací `IVygenerovatObchod`. Má metody, které ze seznamu zboží vytvoří jednotlivé skupiny. Tuto logiku nikterak uživatel ani terapeut neovládá, dochází k ní automaticky v okně `NastaveniForm`, jakmile je vybrána cílová složka s obrázky.

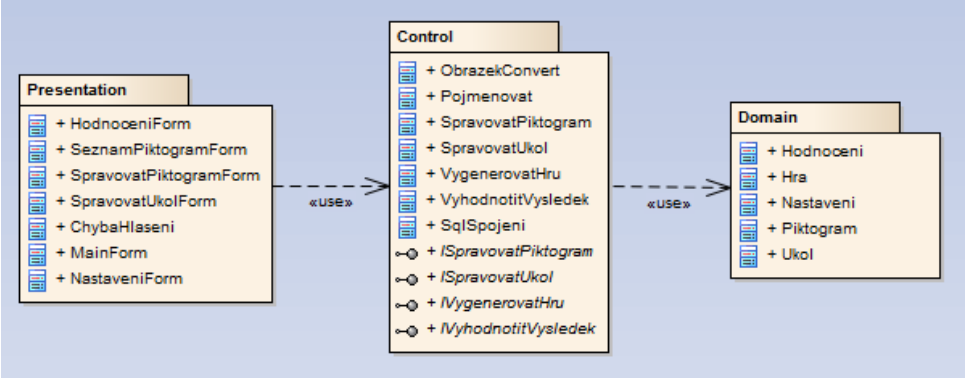
UC002-Vygenerovat úkol



Obrázek 32 – Model tříd případu užití UC002-Vygenerovat úkol. Zdroj: vlastní

Diagram tříd pro UC002-Vygenerovat úkol, tvoří tři entitní třídy, dvě z nich jsou popsány výše, jde o `Zbozi` a `SkupinaZbozi`, dále je to `Nastaveni`. Používá se ke konfiguraci aplikace, má atributy jako je počet zboží, které si musí uživatel zapamatovat, jednotlivé skupiny zboží a seznamy zboží, jež jsou načteny v programu, dále obtížnosti, čas atd. Nad těmito základními třídami pracuje control `VygenerovatUkol` jež je implementací `IVygenerovatUkol`, má za úkol generovat úkol, který je tvořený jednotlivým zbožím, jejž si má uživatel zapamatovat. Zboží je vybráno na základě nastavení, které provedl terapeut, může jít o vybrání jen určitě skupiny a také počtu. Kromě generování zadání, má také metodu pro zobrazení vygenerovaného úkolu, kde jednotlivé zboží je promítáno v intervalu dle nastavení. To má na starost formulář `ZadaniUkoluForm`. Po promítnutí je spuštěna hra a uživatel plní košík zbožím, které mu bylo promítnuto. Generování úkolu se spouští tlačítkem start ve formuláři `MainWindow`.

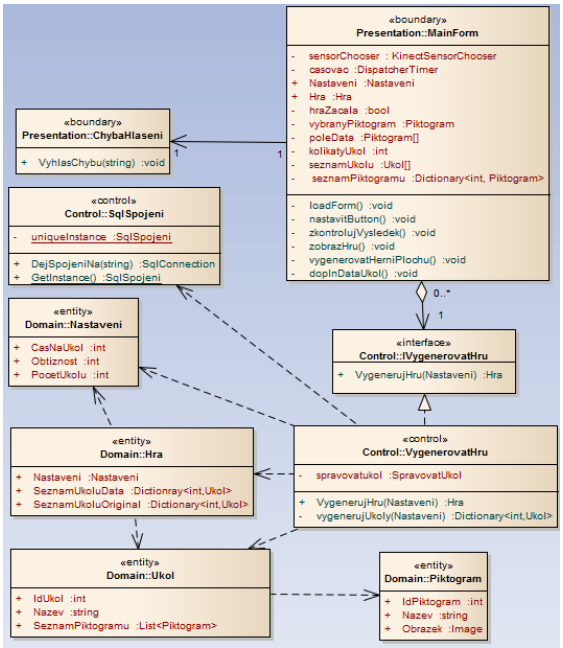
5.5.3 Piktogram modul



Obrázek 33 – MVC architektura aplikace Piktogram modul. Zdroj: vlastní

MVC architektura aplikace Piktogram modul se skládá ze tří vrstev. Zdejším základním kamenem je domain s třídami stereotypu entity, nejvýznamnější jsou `Piktogram` a `Ukol`. Ve vrstvě jenž řídí logiku aplikace je významná třída `VygenerovatHru`, z názvu je patrné, že půjde o vygenerování hry pro uživatele. Dále jednotlivé správy pro úkoly (`SpravovatUkol`) a piktogramy (`SpravovatPiktogram`). Ovládání probíhá pomocí několika formulářů, hlavní okno je `MainForm`, ze kterého se přistupuje k dalším jako je `SpravovatPiktogramForm` nebo `SpravovatUkolForm`. V hlavním okně se také nachází i samotná hra. Chybové hlášky obstarává třída `ChybaHlaseni`. Tato aplikace má vlastní databázi, ve které jsou uloženy jednotlivé úkoly a piktogramy. Pro komunikaci s aplikací se používá `SqlSpojeni`, jež odpovídá návrhovému vzoru Singleton.

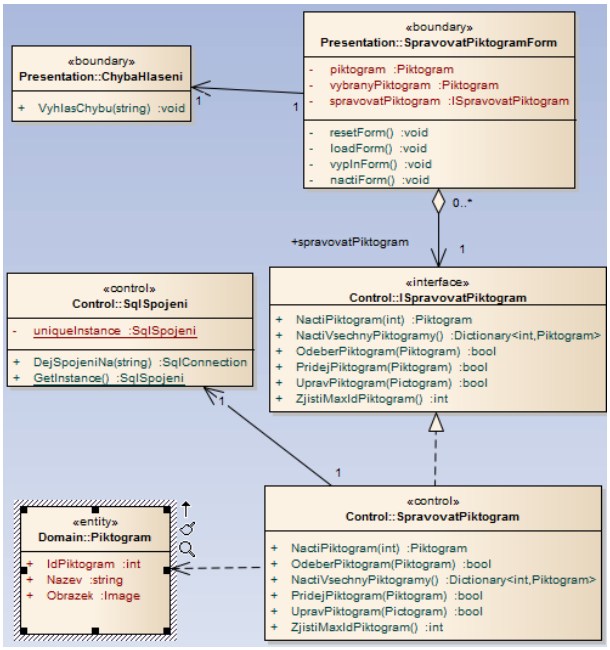
UC006-Vygenerovat Hru



Obrázek 34 – Model tříd případu užití UC006-Vygenerovat Hru. Zdroj: vlastní

Diagram tříd pro model případů užití UC006-Vygenerovat Hru, tvoří několik tříd. Mezi entity tvořící model patří třída `Hra`, ta obsahuje atributy, jako jsou seznamy úkolů, jde o zadání a poté o data co zadal uživatel, dále je zde nastavení aktuální hry. Nastavení (třída `Nastaveni`) hry se týká pouze času, jak dlouho bude trvat maximálně jeden úkol a počtu. Entita `Ukol` má za atributy název, identifikátor a seznam jednotlivých piktogramů v daném pořadí. `Piktogram` obsahuje atributy název, identifikátor a obrázek. Nad těmito entitami pracuje control `VygenerovatHru`, který je implementací rozhraní `IVygenerovatHru`, obsahuje metodu pro vygenerování hry, kde parametrem je nastavení zadané terapeutem. Veškeré úkoly jsou načteny z databáze a poté je z nich vybrán náhodně daný počet. Generování hry se spustí ve formuláři `MainForm` tlačítkem start.

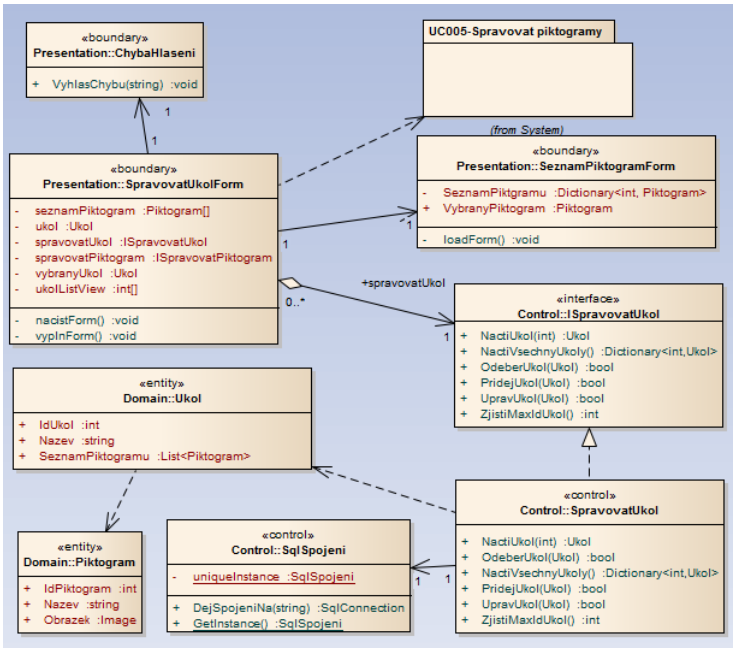
UC005-Spravovat piktogramy



Obrázek 35 – Model tříd případu užití UC005-Spravovat piktogramy. Zdroj: vlastní

Realizaci analytického modelu tříd případu užití pro UC005-Spravovat piktogramy vytváří control třída `SpravovatPiktogram`, jež je implementací rozhraní `ISpravovatPiktogram`, řídí veškerou správu nad piktogramy v aplikaci. Obsahuje metody na přidávání, načítání, odebírání atd. Všechny tyto metody používají třídu `Piktogram`, která byla popsána výše. Pro komunikaci s databází se využívá `SqlSpojeni`. Veškeré operace provádí terapeut pomocí formuláře `SpravovatPiktogramForm`. Vytváření piktogramů probíhá nastavením názvu a vybráním zdrojové cesty k obrázku ve formátu png. Veškeré obrázky jsou poté uloženy přímo do databáze jako binární data pro zajištění jednoduché přenositelnosti.

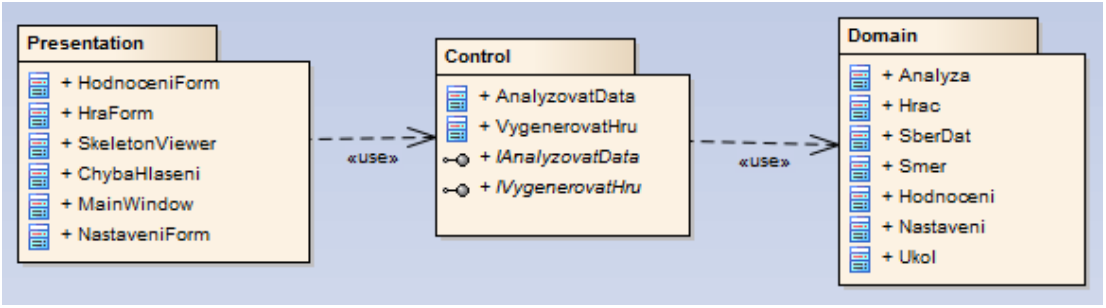
UC007-Spravovat úkoly



Obrázek 36 – Model tříd případu užití UC007-Spravovat úkoly. Zdroj: vlastní

Samotná správa piktogramů nevytváří jednotlivé úkoly, proto se používá správa úkolů. Jednotlivé úkoly jsou tvořeny seznamem piktogramů v daném pořadí. Pro správu se používá formulář `SpravovatUkolForm`, zde se může přidávat nový úkol, odebírat, upravování změní pouze název úkolu, pokud je třeba změnit piktogramy, je nutné odebrat a znovu zadat celý úkol. U přidávání se vygeneruje počet políček, kam se umisťuji piktogramy, vybírané ze `SeznamPiktogramForm`. Veškerou logiku řídí `SpravovatUkol` jenž je implementací rozhraní `ISpravovatUkol`. Obsahuje jednotlivé metody s operacemi, přidej, odeber, uprav atd. Základem jsou třídy `Ukol` a `Piktogram`, nad kterými se operace provádí. Práce s databází probíhá skrze `SqlSpojeni`.

5.5.4 Chůze modul

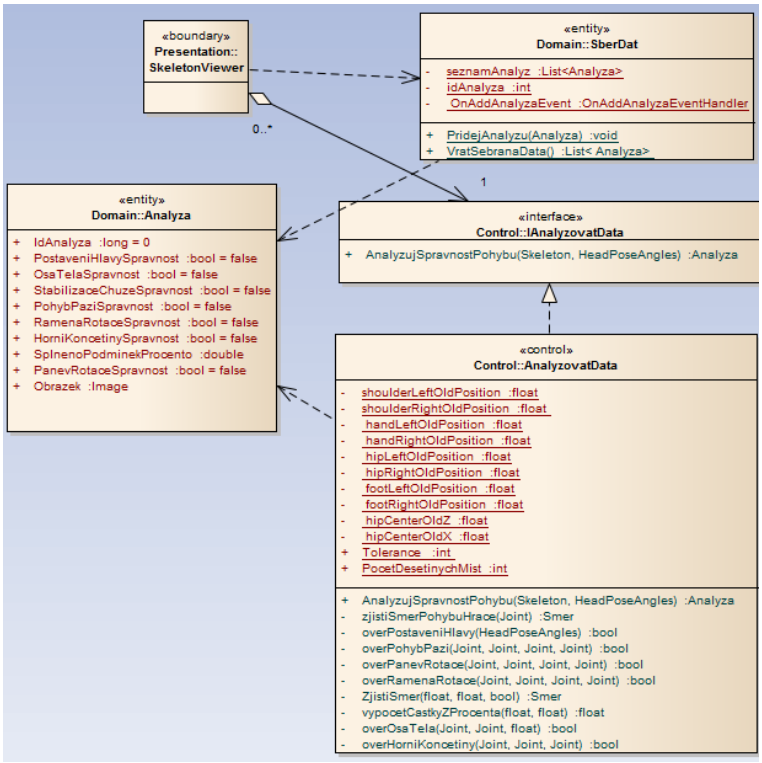


Obrázek 37 – MVC architektura aplikace Chůze modul. Zdroj: vlastní

Aplikace chůze modul je navržena dle architektury MVC, obsahuje tři vrstvy jako předešlé aplikace. Control má dvě důležité třídy a to `AnalyzovatData`, která zpracovává data získaná zařízením Kinect a tvoří analýzu s podmínkami, jež jsou splněny. Druhá je

`VygenerovatHru`, generuje úkol pro uživatele, jakmile splní podmínky z 80%. Vrstva Presentation, obsahuje formuláře pro ovládání aplikace. Nejzajímavější je `MainWindow`, který ukazuje aktuálně splněné podmínky, zde informace pozoruje terapeut, dále `HraForm`, tu používá uživatel. Vše využívá třídy z vrstvy domain a nejdůležitější je zde `Analyza`.

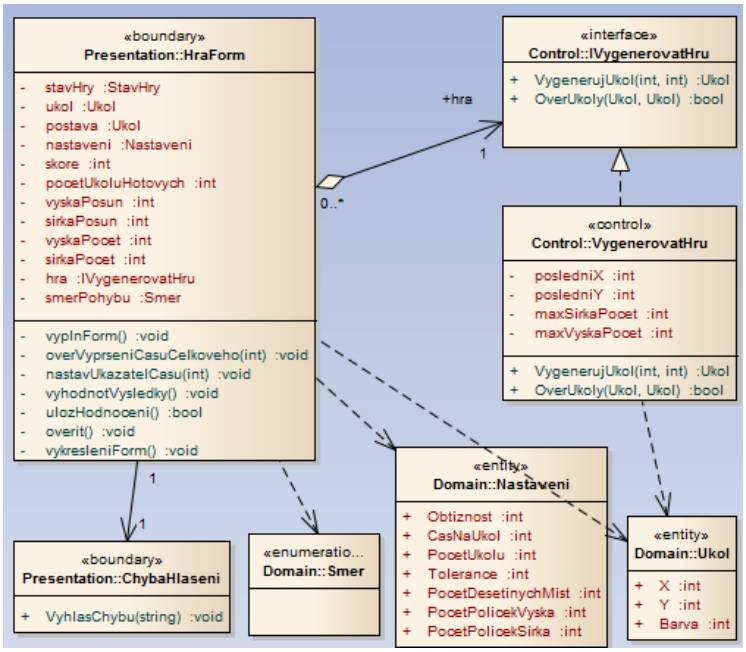
UC002-Analyzovat pohyb



Obrázek 38 – Model tříd případu užití UC002-Analyzovat pohyb. Zdroj: vlastní

Analytický model tříd pro UC002-Analyzovat pohyb, je složen z entit `Analyza`, která má atributy identifikátor, snímek, procentuální splnění, jednotlivé podmínky potřebné pro stabilizaci chůze. `SberDat`, jež ukládá analýzy splňující stoprocentní stabilní chůzi. Analýzu dělá control `AnalyzovatData`, který je implementací rozhraní `IAnalyzovatData`. Tato třída má několik metod, hlavní a jediná veřejná je `AnalyzujSpravnostPohybu`, která má za parametry skeletona získaného ze zařízení Kinect a pózu hlavy. Pro jednotlivé podmínky analýzy jsou samostatné operace testující data a rozhodnou o splnění, zda je splněno nebo ne. Poté dojde k otestování procentuálního plnění, pokud je nad 80%, zjistí směr, jakým se uživatel posune. Analýza probíhá v prvku patřícího do vrstvy presentation a to `SkeletonViewer`.

UC005-Vygenerovat hru



Obrázek 39 – Model tříd případu užití UC005-Vygenerovat hru. Zdroj: vlastní

Diagram tříd pro model případu užití UC005-Vygenerovat hru, základ tvoří několik entit, jde o `Nastaveni` a `Ukol`. `Nastaveni` obsahuje samotné nastavení aplikace, jež volí terapeut před spuštěním hry, jde o atributy jako je počet úkolů, čas na jeden úkol, velikost herní plochy, tolerance a počet desetinných míst pro analýzu přesnosti pohybu. `Ukol` je vygenerovaný úkol, který plní uživatel, jeho atributy jsou souřadnice X a Y, dále pak barva. Logiku tvořící tento případ užití řídí control `VygenerovatHru`, jenž je implementací rozhraní `IVygenerovatHru`. Generuje úkoly pro uživatele pomocí metody `VygenerujUkol` s parametry udávající souřadnice uživatele, aby se úkol nenacházel na místech, kde stojí. Druhou metodou je `OverUkoly` pro ověření zda uživatel se nachází na místě, kde je úkol. Generování hry probíhá v okně `HraForm`, která je vytvořena dle nastavení. Kromě zmíněných entit používá také enumerační typ `Smer`, určující kam se uživatel posune (`Dopredu`, `Dozadu`, `Doleva`, `Doprava`).

5.6 Použité návrhové vzory

Cílem návrhových vzorů je získání dobrého objektově orientovaného designu. Jde o obecné řešení často se opakujících návrhových problémů. Vzory nejsou teoretické konstrukce, ale jedná se o sbírku dlouhodobých zkušeností profesionálních návrhů. Při návrhu je nutné neustále posuzovat, zda se k dané situaci dá použít některý z návrhových vzorů. Pokud ano, tak se problém vyřeší podle vzoru. Návrhové vzory nejsou hotové knihovny, které by už obsahovaly hotová řešení, tedy nejde je přímo zapojit do kódu. Ale je nutné si je dobře představit v hlavě, protože jde o způsob myšlení. [22]

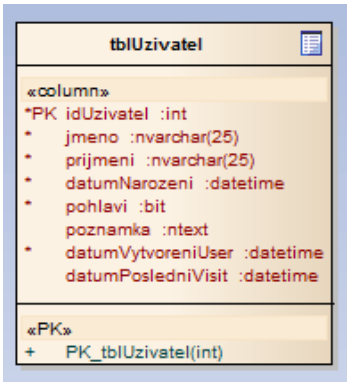
5.6.1 Singleton

Jde o návrhový vzor, používaný v této práci. Využívá se při řešení problému, kdy je potřeba, aby v celém programu běžela pouze jedna instance třídy. Tento návrhový vzor zabezpečí, že třída bude mít jedinou instanci a poskytne k ní globální přístupový bod. Singleton (jedináček) je také často využíván jako součást jiných návrhových vzorů jako jsou například Fasáda. Využití v této práci je u připojování k databázi. Každé aplikaci stačí přesně jedno připojení. Zde je nejčastější problém, že správa připojení se typicky zanedbává. Běžná aplikace má otevřené desítky připojení. Počet připojení k databázovému serveru je omezený. Veliký počet otevřených připojení může způsobit problémy s výkonem. Implementace spočívá ve společném soukromém konstruktoru, který zaručí, že nedojde k vytvoření další instance. Požadovaná instance se vytvoří uvnitř třídy a její odkaz se uloží do statického atributu. Jednotlivé varianty implementace se pak odlišují tím, kdy a jak se objekt konstruuje a jak jej mohou ti ostatní získat. [23]

5.7 Datový model

Datové modelování představuje jednu ze základních součástí analýzy každého softwarového projektu, tedy i projektu, jehož cílem je vytvořit aplikaci uchovávající data v databázi. Správný návrh datové struktury může do značné míry ovlivnit bezporuchovost, udržovatelnost a rozšiřitelnost výsledné aplikace. Při datovém modelování se vytváří nejprve konceptuální datový model. Konceptuální datový model představuje určité zobecnění oproti konkrétní implementaci datové struktury v relační, objektové, případně nativní XML databázi. Zobecněním se získá nezávislost modelu na konkrétním databázovém systému, ale zároveň schopnost tento model kdykoliv převést do konkrétního implementačního prostředí. Datový model reprezentuje informace zpracovávané a v aplikaci. Jedná se o konceptuální model, který je velice podobný klasickým ER diagramům používaným pro návrh struktury relačních databází. Model je tvořen sadou entit, které jsou mezi sebou provázány pomocí vazeb s příslušnou kardinalitou (násobností). Pomocí analytického modelu tříd lze určit, jaké aplikace bude využívat data. Podle toho je třeba navrhnout tabulky, stanovit vztahy mezi nimi, určit primární klíče. [24]

5.7.1 Kinect Therapy Starter



Obrázek 40 – Datový model aplikace Kinect Therapy Starter. Zdroj: vlastní

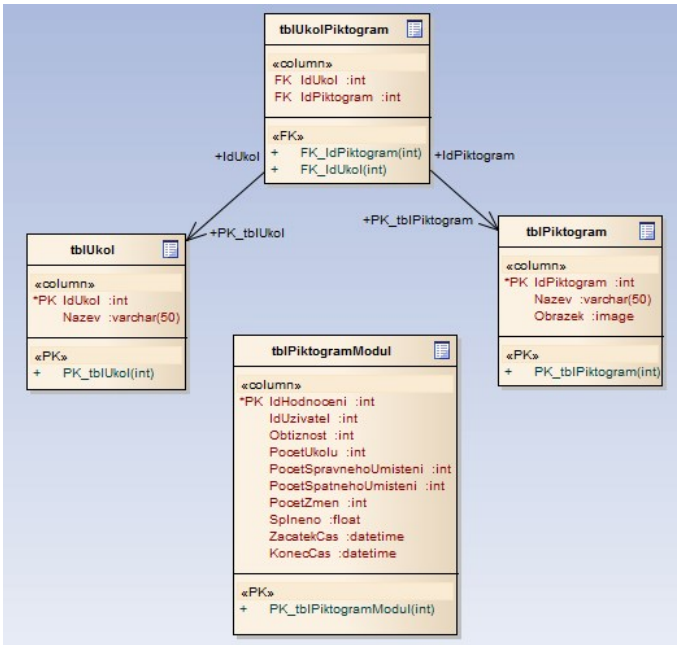
Výše uvedený obrázek představuje datový model aplikace Kinect Therapy Starter. Je tvořen jedinou tabulkou. Jde o `tblUzivatel`, jež uchovává základní informace o uživateli. V aplikaci se předpokládá, že u uživatele se může změnit příjmení, jméno případně i další hodnoty. Mezi vedlejší atributy patří pohlaví, kde se jedná o typ bit, hodnota nula znamená, že jde o muže, jednička ženu. Dalším vedlejším atributem je poznámka. Dále jsou zde uvedená data narození uživatele, registrace, poslední návštěvy, slouží terapeutovi pro informativní přehled. Primárním klíčem tabulky je `IdUzivatel`, neboli unikátní identifikátor uživatele. S tímto klíčem dále pracují jednotlivé moduly.

5.7.2 Obchod modul

Obrázek 41 – Datový model aplikace Obchod modul. Zdroj: vlastní

Datový model pro aplikaci Obchod modul obsahuje jedinou tabulku a to `tblObchodModul`. Jde o tabulku s hodnocením hry pro daného uživatele. Je tvořena atributy z hodnocení získané během celé jedné hry, může jít o počet vložení, odebrání, což jsou charakteristické rysy výsledků pro danou aplikaci. Tabulka musí splňovat normu pro správnou funkčnost a dynamičnost celkového systému, proto obsahuje standartní sloupce, jako jsou `IdHodnoceni`, `IdUzivatele`, `Obtiznost`, `Splneno`, `ZacatekCas`, tyto atributy využívá centrální aplikace, jež zpracovává celkové hodnocení pro daný modul. Primárním klíčem tabulek modulů je vždy `IdHodnoceni`.

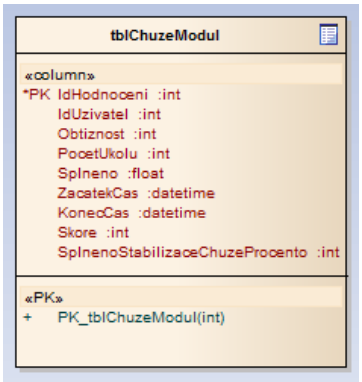
5.7.3 Piktogram modul



Obrázek 42 – Datový model aplikace Piktogram modul. Zdroj: vlastní

Pro aplikaci Piktogram modul je datový model tvořen několika tabulkami. První je zaměřená na hodnocení jako u ostatních modulů, jde o tabulku `tblPiktogramModul`. Obsahuje atributy určující výsledné hodnocení pro tento modul, jde o počet správně splněných úkolů, dále má standartní sloupce, pro splnění standartu. Jako další tabulky jsou zde `tblUkol`, `tblPiktogram` a propojovací `tblUkolPiktogram`. Tyto tři jsou součástí přímo aplikace, `tblUkol` uchovává jednotlivé úkoly, jejím primárním klíčem je `IdUkol`, dále je zde název úkolu. `TblPikogram` obsahuje jednotlivé piktogramy nahrané uživatelem, primární klíč je `IdPiktogram`, významným atributem je `Obrazek`, který je typu Image. Jelikož úkol je tvořen několika piktogramy a piktogramy jsou v několika úkolech, je vazba (kardinalita vztahu) typu M:N, pro rozložení vazby na 1:N je vytvořena spojovací tabulka `tblUkolPiktogram`, jež obsahuje `IdUkol` a `IdPiktogram`, což jsou cizí klíče. Má také nastaveno kaskádové pravidlo u odstraňování a upravování, což zajistí automatickou správu propojení.

5.7.4 Chůze modul



Obrázek 43 – Datový model aplikace Chůze modul. Zdroj: vlastní

Datový modul pro aplikaci Chůze modul je tvořen jednou tabulkou `tblChuzeModul`, která slouží pro uchování hodnocení. Jsou zde atributy charakterizující výsledky aplikace, jde například o skóre, jež uživatel získal při hraní, SplnenoStabilizaceChuzeProcento určující kolikrát byla splněna stoprocentní stabilizace chůze, počet úkolů celkem. Dále obsahuje standartní sloupce potřebné pro centrální aplikaci, opět jde o `IdHodnoceni`, `IdUzivatel`, `Obtiznost`, `Splneno`, `ZacatekCas`. Primárním klíčem je `IdHodnoceni`.

6 Implementace systému

Implementace systému je proces uskutečňování teoreticky stanovené myšlenky nebo projektu za účelem jejího dalšího použití. Implementaci předchází analýza zadání, plánování postupu a očekávaných výsledků, jež bylo popsáno v předešlé kapitole o návrhu systému. Nesoulad mezi předpokladem a skutečností může být způsoben chybou implementace nebo chybou samotné metody. Správná metoda je funkční nezávisle na způsobu implementace. Tato kapitola se zabývá popisem nejdůležitějších části tvorby jednotlivých aplikací. První podkapitola rozebírá technologie použité pro implementování systému. Druhá obsahuje popis požadavků nutných pro používání systému. Poté jsou rozebrány implementace jednotlivých aplikací systému. [25]

6.1 Technologie pro implementaci systému

6.1.1 Microsoft Visual Studio 2012

Jedná se o balík nástrojů a služeb určený k vývoji softwarových aplikací pro desktopové i dotykové prostředí Windows, pro web/HTML 5, SharePoint, mobilní zařízení i cloudové prostředí. Umožňuje vývoj všech typů aplikací pro prakticky všechny druhy koncových zařízení. Od mobilu až po cloud. V rámci MSDN služeb obsahuje nástroje podpora řízení softwarových projektů, testování a spolupráce v týmu. Existuje mnoho edic, Express pro začátečníky, Test pro testery až po Ultimate pro opravdové vývojáře a team leadery. Pro jednotlivé aplikace systému je použita verze Visual Studio Professional, která je pro studenty zdarma. [26]

6.1.2 C sharp - C#

Aplikace systému jsou programovány ve vysokoúrovňovém objektově orientovaném programovacím jazyku C# (C Sharp) od společnosti Microsoft, jež je součástí nástroje Microsoft Visual Studio. Tento jazyk má nesmírnou výhodu, že pracuje na platformě .NET Framework, která již dnes je součástí operačních systémů Windows Vista,7 nebo vyšší. Lze jej také doinstalovat i na starší systémy, jako například Windows XP jenž se i v dnešní době velmi používají. Tato technologie umožňuje přenositelnost naprogramované aplikace bez nutnosti nestandardních prerekvizit a její velikost i při použití formulářových prvků zabírá několik KB (kilobajtů). [27]

6.1.3 Windows Presentation Foundation – WPF

Jde o podmnožinu .NET Frameworku od verze 3.0, který používá značkovací jazyk XAML pro vytvoření uživatelsky bohatého rozhraní (RUI). Technologie WPF je vestavěná do Windows Vista, Windows 7 a vyšších, je i stažitelná pro Windows XP SP2. Díky XAMLu jsou od sebe odděleny funkčnost a vzhled aplikace. Cílem WPF je sjednotit poutavé uživatelské rozhraní, 2D a 3D grafiku, vektorovou a rastrovou grafiku, animace, vázání dat a audio a video. WPF je tedy přímým konkurentem z vlastní stáje pro starší Windows formuláře. Veškerá grafika včetně samotných WPF oken funguje pomocí Direct3D knihoven. Což umožňuje hardwarovou akceleraci pomocí grafické karty a pokročilejší grafické schopnosti. [28]

6.1.4 Microsoft Sql Compact 4.0

Jde o bezplatnou databázi umožňující vývojářům softwaru vytvářet weby ASP.NET a aplikace pro systém Windows. SQL Server Compact 4.0 má nízké nároky na místo a podporuje soukromé nasazení binárních souborů ve složce aplikace, snadný vývoj aplikací v sadě Visual Studio a službě WebMatrix a bezproblémovou migraci schématu a dat do systému SQL Server. Databázový systém umožňuje pracovat v prostředích se středním nebo částečným vztahem důvěryhodnosti na webových serverech a lze jej snadno nasadit společně s webem u poskytovatelů služeb webového hostingu třetích stran. Nasazení systému SQL Server Compact je dále zjednodušeno vlastnictvím všech požadovaných spravovaných součástí a knihoven DLL pro platformy x86 a x64, včetně knihoven runtime Visual C++ 2008 v jedné soukromé složce v umístění instalace systému SQL Server Compact. Všechny aplikace v této práci používají tuto databázi jak pro uložení hodnocení, tak i pro ostatní data. [29]

6.1.5 SQL Server Compact Toolbox

Jedná se o doplněk, který do nástroje Microsoft Visual Studio přidává několik funkcí na správu všech aspektů databází systému Microsoft SQL Server Compact. Umožňuje skriptování tabulek a údajů, import ze serveru SQL Server, CSV souborů a dalších. Zajímavou funkcí je také možnost převodu mezi jednotlivými verzemi SQL Server Compact, například z verze 3.5 na 4.0, nebo zpětně, bez ztráty dat.

6.1.6 Kinect for Windows SDK 1.7

Softwarový vývojový kit pro zařízení Kinect for Windows. Umožňuje vývojářům používat programovací jazyky C++, C# nebo Visual Basic k vytvářením aplikací, která podporují gesta a rozpoznávání hlasu pomocí Kinect senzor. SDK nabízí nepřeberné množství nástrojů, které pomáhají vývojářům zjednodušit vývoj aplikací a vytvářet inteligentnější aplikace. [13]

6.1.7 Kinect for Windows Developer Toolkit 1.7

Nástroj Toolkit obsahuje Kinect Fusion, který přináší nové ovládací prvky pro práci s jednoduchými gesty. API, jež ušetří hodiny času vývoje. Kromě nástrojů obsahuje ukázky zdrojových kódů a další zdroje, které pomohou zefektivnit bezdotykový vývoj aplikací na Kinect for Windows. V této práci jsou použity knihovny z toho nástroje (Microsoft.Kinect.Toolkit), přesněji jde o knihovny na rozpoznání hlavy (Microsoft.Kinect.Toolkit.FaceTracking), zobrazení skeletona, dále ovládací prvky (Microsoft.Kinect.Toolkit.Controls) jako jsou tlačítka, nebo posuvníky, jež reagují na Kinect senzor.[13]

6.2 Požadavky

V této části jsou popsány nutné součásti pro správný chod aplikací systému. Pokud některá z nich nebude nainstalovaná na počítači, kde bude systém používán, tak jednotlivé aplikace nepůjdou spustit. První a nejdůležitější podmínkou je mít nainstalovaný operační systém Windows 7 nebo vyšší. Další z podmínek jsou uvedeny níže.

6.2.1 .Net Framework 4

Jedná se o platformu, nebo také rozhraní, které je komplexní a konzistentní programovací model společnosti Microsoft určený pro sestavování aplikací, které se vyznačují vizuálně působivým uživatelským rozhraním, bezproblémovou a zabezpečenou komunikací a schopností modelovat řadu obchodních procesů. Rozhraní .NET Framework 4 funguje společně se staršími verzemi rozhraní Framework. Aplikace, které jsou založeny na starších verzích rozhraní Framework, bude možné nadále spouštět v novějších verzích. Všechny aplikace jsou naprogramované pro verzi .Net Framework 4, bez ní není možné žádnou z nich spustit.

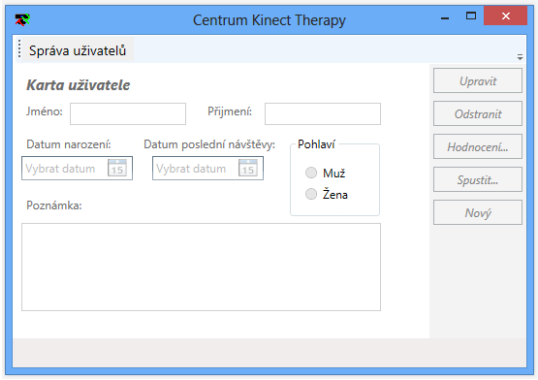
6.2.2 Microsoft SQL Server Compact 4.0

Tento databázový systém byl již popsán v předešlé kapitole, ale je nutné ho mít nainstalovaný i u počítače, kde systém je nainstalován a používán. Při instalaci je potřeba vzít správnou verzi, tedy 4.0, ale také instalátor, jež je pro správnou architekturu operačního systému (x86, x64).

6.2.3 Kinect for Windows Runtime 1.7

Kinect for Windows Runtime zahrnuje pouze ovladače pro zařízení Kinect for Windows a runtime prostředí. Instaluje se na koncové počítače, kde se používají aplikace využívající senzor. Neobsahují žádné vývojové knihovny. Runtime požaduje minimálně počítač s procesorem Dual-core 2.66-GHz, rozhraním USB 2.0 a 2 GB RAM.

6.3 Kinect Therapy Starter



Obrázek 44 – Ukázka formuláře aplikace Kinect Therapy Starter. Zdroj: vlastní

Veškeré formuláře jsou tvořeny pomocí značkovacího jazyku XAML používaného ve windows presentation foundation, jeho výhody byly již popsány. Na ukázku je zde XAML kód na vytvoření levého menu, které je na obrázku výše. Tři tečky v názvu symbolizují, že po kliknutí se objeví dialogové okno.

```
<StackPanel Orientation="Vertical" Name="controlPanel">
     <Button Name="upravitButton"
Click="upravitButton_Click">Upravit</Button>
     <Button Name="odstranitButton"
Click="odstranitButton_Click">Odstranit</Button>
```

```
     <Button Name="vyhodnotitButton"
Click="vyhodnotitButton_Click">Hodnocení...</Button>
     <Button Name="spustitButton"
Click="spustitButton_Click">Spustit...</Button>
     <Button Name="novyButton"
Click="novyButton_Click">Nový</Button>
</StackPanel>
```

Pro správu uživatelů se používá třída `SpravovatUzivatele`, obsahuje metody pro vytváření, načtení atd. Pracuje tedy s tabulkou `tblUzivatel`. Pro ukázku je zde uveden kód metody pro načtení uživatele, dle zadaného identifikátoru.

```
public Uzivatel NactiUzivatele(int idUser){
try{
     string qryNactiUzivatele = "SELECT * FROM " +
tblUzivatel + " WHERE " + clIdUzivatel + "=" + idUser;
     SqlCeConnection sqlSpojeni =
     SqlSpojeni.GetInstance().DejSpojeniNa(SqlSpojeni.SqlServ
     erZdroj);
     sqlSpojeni.Open();
     DataSet zaznamy = new DataSet();
     SqlCeDataAdapter sqlAdapter = new SqlCeDataAdapter();
     sqlAdapter.SelectCommand = new
     SqlCeCommand(qryNactiUzivatele, sqlSpojeni);
     sqlAdapter.Fill(zaznamy, tblUzivatel);
     sqlSpojeni.Close();
     Uzivatel uzivatel = null;
     if (zaznamy.Tables[tblUzivatel].Rows.Count > 0){
          uzivatel = new Uzivatel(){
          IdUzivatel = idUser,
          Jmeno =
     zaznamy.Tables[tblUzivatel].Rows[0][clJmeno].ToString(),
          Prijmeni =
     zaznamy.Tables[tblUzivatel].Rows[0][clPrijmeni].ToString
     (),
          DatumNarozeni =
     DateTime.Parse(zaznamy.Tables[tblUzivatel].Rows[0][clDat
     umNarozeni].ToString()),
          Pohlavi =
     (bool)zaznamy.Tables[tblUzivatel].Rows[0][clPohlavi],
          DatumPosledniVisit =
     DateTime.Parse(zaznamy.Tables[tblUzivatel].Rows[0][clDat
     umPosledniVisit].ToString()),
          DatumVytvoreniUser =
     DateTime.Parse(zaznamy.Tables[tblUzivatel].Rows[0][clDat
     umVytvoreniUser].ToString()),
          Poznamka =
     zaznamy.Tables[tblUzivatel].Rows[0][clPoznamka].ToString
     ()
          };
```

```
     }
     return uzivatel;
}catch{
     return null;
}
}
```

Metody pro operace s databází jsou vesměs podobné. V prvé části se definuje dotaz, do kterého se zakomponují hodnoty z parametru metody. Vytvoří se spojení s databází a následně se začínají lišit operace. Sql operace mohou být několika typů a to buď dotazovací, nebo nedotazovací. V této ukázce je dotaz dotazovací na získání dat z databáze, ve formě ADO.NETu. Využívá se zde typ `DataSet`, který uchová získaná data v paměti, bez nutnosti neustálého připojení. Aby `DataSet` mohl být naplněný, používá k tomu třídu `SqlDataAdapter`, ta představuje sadu příkazů, připojení k databázi. Dále i umožňuje aktualizovat databázi serveru SQL Server. Po získání `DataSetu` dojde k vybrání tabulky a jejímu dalšímu zpracování, v této ukázce se prochází jednotlivé řádky a vytváří se z nich třída `Uzivatel`. Druhým typem Sql operací jsou nedotazovací, tedy nezískávají data, ale jen je modifikují. Může jít o Insert, Update, Delete, vezmou dotaz, po připojení vytvoří `SqlCommand` a příkazem `ExecuteNonQuery()` ho vykonají. Tento způsob, ale neodpovídá technice ADO.NET.

V centrální aplikace je nutné načíst moduly, tedy aplikace, které jsou v systému. K tomu slouží třída `NacistModul`. Jejím cílem je z dané složky, v tomto případě „config“ načíst všechny konfigurační XML soubory a z nich vytvořit seznam modulů. Níže je ukázka z metody na parsování XML souboru a tedy vytvoření entity `ConfigModul`.

```
private ConfigModul parseXmlSoubor(string soubor, int id){
ConfigModul configModul = null;
using (DataSet dataset = new DataSet()){
try{
     FileStream fs = new FileStream(soubor, FileMode.Open,
FileAccess.Read, FileShare.Read);
     XmlTextReader xmlR = new XmlTextReader(fs);
     dataset.ReadXml(xmlR);
     foreach (DataRow radek in dataset.Tables[0].Rows){
          configModul = new ConfigModul();
          configModul.IdModulu = id;
          configModul.NazevModulu =
     (string)radek[xmlNazevModulu];
          configModul.CestaKModulu =
     (string)radek[xmlCestaKModulu];
          configModul.SqlTabulkaModulu =
     (string)radek[xmlSqlTabulkaModulu];
          configModul.PopisModulu =
     (string)radek[xmlPopisModulu];
          try{
```

```
                TypModulu typModulu =
            (TypModulu)Enum.Parse(typeof(TypModulu),
            radek[xmlTypModulu].ToString());
                configModul.TypModulu = typModulu;
            }catch{
                ChybaHlaseni.VyhlasChybu(ChybaHlaseni.CHYBA_XM
            L_TYPMODULU + "\n" + soubor + "\n[Pohyb, Logika,
            Pamet]");
                return null;
            }
        }
}catch{
        return null;
}
}
return configModul;
}
```

Metoda má jako parametr cestu k soboru a identifikátor modulu. Pro práci s XML strukturou se použije již zmíněny `DataSet`, má totiž metodu `ReadXml`, pro přečtení xml souboru, což z něj vytvoří klasickou tabulku. Daný soubor načte třídou `FileStream`, která slouží pro práci se streamy, v tomto případě jde o práci se soubory. Poněvadž jde o XML, tak pro jeho přečtení se používá třída `XmlTextReader`. Do jejího konstruktoru se zadá stream a následně je přečten metodou `ReadXml`. Poté se prochází tabulka vytvořená z XML v `DataSetu`. Z jejích řádků se vytváří třída `ConfigModul`, problém může nastat u typu modulu, smí být pouze `Logika`, `Pohyb` nebo `Pamet`. Správný konfigurační XML soubor má tento tvar. Ukázka XML obsahuje konfigurační údaje o aplikace Obchod modul.

```
<?xml version="1.0" encoding="windows-1250" ?>

<ConfigModul>

        <NazevModulu>Obchod</NazevModulu>

        <CestaKModulu>.\ObchodModul\ObchodModul.exe</CestaKModulu>

        <SqlTabulkaModulu>tblObchodModul</SqlTabulkaModulu>

        <PopisModulu>Trénuje krátkodobou paměť.</PopisModulu>

        <TypModulu>Pamet</TypModulu>

</ConfigModul>
```

6.4 Obchod modul

Obrázek 45 – Ukázka formuláře aplikace Obchod modul. Zdroj: vlastní

Aplikace je tvořena formuláři udělané pomocí windows presentation foundation. Obrázek výše ukazuje hlavní okno. Je zde Menu v horní části a poté v dolní části jsou tlačítka pro hru. Uprostřed je dynamicky tvořený panel, obsahující jednotlivé skupiny zboží. Dynamické generování prvků je použito i u dalších aplikací. Nikdy předem systém neví, kolik prvku tvořených z dat bude. Níže je ukázána metoda pro tvorbu tlačítka, jež po kliknutí zobrazí formulář s vybranou skupinou zboží.

```
private KinectTileButton vytvorTlacitko(SkupinaZbozi
skupinaZbozi){
try{
     KinectTileButton tlacitko = new KinectTileButton();
     tlacitko.Width = 300;
     tlacitko.Height = 300;
     StackPanel panelTlacitka = new StackPanel();
     panelTlacitka.Children.Add(new Image() { Source =
skupinaZbozi.ObrazekSkupiny.Source, Height = 200, Width = 200
});
     panelTlacitka.Children.Add(new Label() { Content =
skupinaZbozi.NazevSkupiny, VerticalContentAlignment =
VerticalAlignment.Center, HorizontalContentAlignment =
HorizontalAlignment.Center });
     tlacitko.Name =
Pojmenovat.VytvorNazev(skupinaZbozi.NazevSkupiny);
     tlacitko.Background = Brushes.White;
     tlacitko.Content = panelTlacitka;
     tlacitko.Click += new
System.Windows.RoutedEventHandler(tlacitko_Click);
     return tlacitko;
}catch{
     return null;
}
}
```

V metodě `vytvorTlacitko` se předává parametr, určující o kterou skupinu zboží jde. Třída `SkupinaZbozi` obsahuje název, obrázek a panel se zbožím. Vytvořené tlačítko má mít v sobě obrázek a popisek tvořený daty z této třídy. Jelikož aplikace má mít možnost ovládání přes kinect senzor jde o typ `KinectTileButton`, získaný ze třídy `Microsoft.Kinect.Toolkit.Controls`. Aby se udržel standartní vzhled, tak uvnitř tlačítka je `StackPanel`, který obsahuje prvek `Image` a `Label`, jež jsou poskládány za sebou. K přidání prvků slouží metoda `Add` ve vlastnosti `Children` u panelu. Do těchto prvků se přiřadí hodnoty ze třídy `SkupinaZbozi`. Dalším krokem je vytvoření názvu tlačítka, k tomu se využije třída `Pojmenovat`, která zajistí standartní název objektu. Zde je použit jako pojmenování název skupiny zboží. Nyní je třeba nastavit pro tlačítko událost, která zajistí funkčnost po kliknutí. Ukázka operace po kliknutí na tlačítko je níže.

```
void tlacitko_Click(object sender,
System.Windows.RoutedEventArgs e)
{
KinectTileButton clickedButton = (KinectTileButton)sender;
SkupinaZbozi vybranaSkupinaZbozi =
nastaveni.SkupinyZbozi.First(x =>
Pojmenovat.VytvorNazev(x.Key) == clickedButton.Name).Value;

using (SkupinaZboziForm skupinaZboziForm = new
SkupinaZboziForm(vybranaSkupinaZbozi, sensorChooser)){
     skupinaZboziForm.Owner = this;
     skupinaZboziForm.ShowDialog();
}
}
```

Jakmile uživatel klikne na jakékoliv z vygenerovaných tlačítek, dojde ke zpracování události. Prvním krokem je zjištění, které tlačítko bylo zmáčknuté. To se zjistí přes objekt `sender`, který je parametrem metody. Po přetypování na daný typ je možné přistoupit k vlastnostem, jakou je název tlačítka. Protože se název standardizoval, je nutné získat původní název pomocí třídy `Pojmenovat` a její metody `VratPuvodniNazev`. Dle názvu se vyhledá ze seznamu všech skupin zboží tu, co odpovídá názvu. Jakmile ji najde, otevře se dialogové okno pro zobrazení skupiny zboží, které má za parametr vybranou skupinu zboží a senzor kinect. Obdobným principem fungují ostatní metody na dynamické generování ovládacích prvku, dle dat u ostatních aplikací.

Významnou metodou této aplikace je také vytváření zboží z obrázků uložených ve vybrané složce počítače. Jedinou podmínkou je formát jpg. Níže je ukázána metoda pro vytvoření třídy `Zbozi` z obrázku načteného ze složky.

```
private Zbozi vytvorZbozi(string cesta, int idZbozi){
try{
     string[] casti = cesta.Split('\\');
     Zbozi zbozi = new Zbozi(cesta);
```

```
        zbozi.IdZbozi = idZbozi;
        string[] nazevSkupina = casti[casti.Length -
1].Split('_');
        zbozi.Nazev = nazevSkupina[0].Replace('-', ' ');
        zbozi.Obrazek.Name = "obrazek" + idZbozi;
        string skupina = "";
        if (nazevSkupina.Length > 1){
            nazevSkupina[1] = nazevSkupina[1].Substring(0,
        nazevSkupina[1].Length - 4);
            skupina = nazevSkupina[1].Replace('-', ' ');
        }else{
            skupina = "skupina";
            zbozi.Nazev = zbozi.Nazev.Substring(0,
        zbozi.Nazev.Length - 4);
            zbozi.Nazev = zbozi.Nazev.Replace('.', '_');
        }
        zbozi.Skupina = skupina;
        return zbozi;
}catch{
        return null;
}
}
```

Parametrem metody `vytvorZbozi` je cesta k obrázku a identifikátor zboží. První částí je vytvoření instance zboží, kde se do konstruktoru zadává cesta k obrázku. Druhá fáze se zabývá získání názvu zboží, která probíhá pomocí rozdělením řetězce cesty podle daného znaku, v tomto případě lomítka. Rozdělením cesty do pole řetězců je nyní možné získat poslední hodnotu pole, která je například takto „Pomeranč_Ovoce.jpg“. Bohužel toto nestačí jako název a je nutné opět řetězec rozdělit a to podle znaku podtržítka. Tím se získá nové pole složené z názvu a skupiny. Název skupiny obsahuje nechtěné znaky „.jpg“, proto dojde k ořezání. Pokud nastane případ, že neexistuje v názvu obrázku podtržítko, dojde přiřazení neutrální skupiny „skupina“.

6.5 Piktogram modul

Obrázek 46 – Ukázka formuláře aplikace Piktogram modul. Zdroj: vlastní

Jednotlivá okna aplikace (formuláře) jsou vytvářena pomocí windows presentation foundation. Obrázek výše ukazuje hlavní okno, ve kterém probíhá hra. Horní část obsahuje menu s ukazatelem času do konce hry. Spodní část popisky, vybraný piktogram a kontrolu. Střed má dvě části, horní je místo kam dává uživatel piktogramy do správného pořadí, dolní obsahuje rozházené piktogramy, ze kterých uživatel vybírá. V pravé části je šipka pro přesun na další úkol, bez možnosti návratu. Piktogramy, jež se nacházejí ve střední části a prázdná tlačítka s názvem Piktogram + číslo, jsou dynamicky generované prvky, přesněji tlačítka. Princip je stejný jak v kapitole Obchod modul pro vytváření tlačítka.

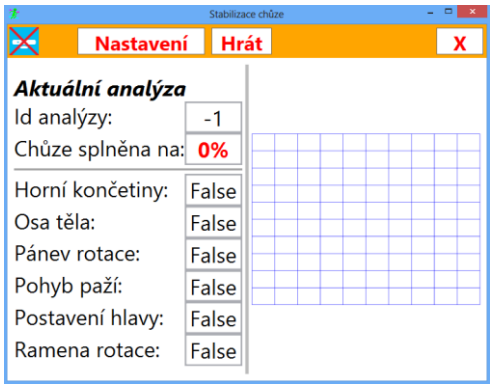
Základní třídou této aplikace je `VygenorovatHru`, jde o stereotyp control a z názvu je již patrné, že má na starost vygenerovat hru pro uživatele. Každá hra obsahuje několik úkolů, ty jsou generovány pomocí soukromé metody `vygenerujUkoly`, která vrátí seznam úkolů. Ukázka této metody je níže:

```
private Dictionary<int, Ukol> vygenerujUkoly(Nastaveni
nastaveni){
     Ukol ukol = new Ukol();
     Dictionary<int, Ukol> vsechnyUkolyDatabaze =
spravovatUkol.NactiVsechnyUkoly();
     Dictionary<int, Ukol> seznamUkoluHry = new
Dictionary<int, Ukol>();
     Random nahoda = new Random();
     int[] poleIdUkolu = new int[vsechnyUkolyDatabaze.Count];
     int index = 0;

     foreach (Ukol ukolRadek in vsechnyUkolyDatabaze.Values){
          poleIdUkolu[index++] = ukolRadek.IdUkol;
     }

     index = 0;
     int idUkol = 0;
     while (index < nastaveni.PocetUkolu){
          idUkol = nahoda.Next(0, poleIdUkolu.Length);
          if
     (!seznamUkoluHry.ContainsKey(poleIdUkolu[idUkol])){
               seznamUkoluHry.Add(poleIdUkolu[idUkol],
          vsechnyUkolyDatabaze[poleIdUkolu[idUkol]]);
               index++;
               System.Threading.Thread.Sleep(100);
          }
     }

     return seznamUkoluHry;
}
```

Operace na generování úkolu má jediný parametr a to je `Nastaveni`, ve kterém je atribut zvaný `PocetUkolu`, tedy určuje, kolik úkolů se má generovat. Prvním krokem je vytvoření prázdného seznamu, který bude poté obsahovat úkoly, zde je zvolen datový typ

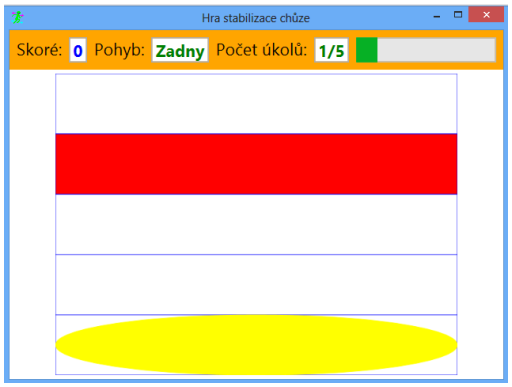
slovník (`Dictionary`), výhoda spočívá v tom, že má klíč, který je libovolný, zde je zvolen datový typ číslo (`int`). Jako klíč se používá identifikátor úkolu a hodnota bude daný úkol. Pomocí klíče se hodnota získá přístupovým časem jedna. Také je nutný seznam úkolů uložených v databázi, který se ihned naplní pomocí třídy pro správu úkolů. Z tohoto seznamu se bude následně náhodně vybírat úkoly. Jelikož identifikátory úkolů nejsou po sobě jdoucí čísla, je nutné mít pole, jež bude obsahovat tyto identifikátory a program bude náhodně generovat index, přes který se k tomuto id dostane. V cyklu `while` probíhá proces generování úkolů, spočívá tedy v náhodném vygenerování indexu ukazujícího na úkolu. Získaný identifikátor se otestuje, zda se nenachází již v seznamu úkolů, pokud ne tak přistoupí k úkolům z databáze a vloží vybraný do seznamu úkolů pro uživatele. Tento proces se opakuje tolikrát, dokud v seznamu úkolu pro uživatele není počet jako dle nastavení.

6.6 Chůze modul



Obrázek 47 – Ukázka formuláře aplikace Chůze modul. Zdroj: vlastní

Veškerá okna jsou vytvořena pomocí windows presentation foundation. Obrázek výše zobrazuje hlavní formulář, který se skládá ze tří částí, horní obsahuje menu, jsou zde tlačítka pro nastavení, spuštění hry uživateli a ukončení aplikace. Střední část je rozdělena právě na zbylé dvě části. Levá obsahuje probíhající analýzu, přesněji stavy podmínek pro stabilizaci chůze, procentuální splnění. V pravé části se nachází mřížka, v níž je reálný obraz z kinect senzoru, dále se zde po rozpoznání postavy objeví i skeleton s podrobným rozpoznáním hlavy. Tento formulář není pro uživatele, jelikož tu neprobíhá žádná hra, k tomu slouží jiný formulář (viz. zobrazení níže), ale slouží terapeutovi, aby pozoroval, kde přesně dělá uživatel chyby. Zda má problém s dodržováním podmínky osy těla, nebo pohybe, paží atd.



Obrázek 48 – Ukázka oka, kde probíhá hra aplikace Chůze modul. Zdroj: vlastní

Další formulář je pro samotnou hru, kterou ovládá uživatel svým pohybem. Jde o herní plochu, kde je náhodně generováno červené políčko a uživatel, neboli žluté políčko, k němu musí dojít. V ukázce je plocha jen o šířce jednoho políčka, výšce pěti políček, toto ovlivňuje terapeut v nastavení aplikace. Pohyb uživatele je přenášen jen tehdy, když splní stabilizaci chůze na osmdesát procent. Poté je posun ve směru, kam se přesunul uživatel. Jakmile je na červeném políčku, zvýší se mu skóre a vygeneruje nové červené políčko. Vše je omezeno časem, který je zobrazen v horní části s dalšími ukazateli.

Nejdůležitější třídou této aplikace je `AnalyzovatData` s veřejnou metodou `AnalyzujSpravnostPohybu`, která má za parametry skeleton a pózu hlavy, oba parametry jsou získány ze zařízení kinect. Jejím cílem je získat ze získaných dat třídu `Analyza`, tedy aktuální analýzu, v níž jsou atributy jako podmínky a procentuální plnění.

```
public Analyza AnalyzujSpravnostPohybu(Skeleton skeleton,
HeadPoseAngles headPose){
     Analyza aktualniAnalyza = new Analyza();
     aktualniAnalyza.HorniKoncetinySpravnost =
overHorniKoncetiny(skeleton.Joints[JointType.HandLeft],
skeleton.Joints[JointType.HandRight],
skeleton.Joints[JointType.Spine]);
     aktualniAnalyza.OsaTelaSpravnost =
overOsaTela(skeleton.Joints[JointType.Head],
skeleton.Joints[JointType.Spine],
vypocetCastkyZProcenta(skeleton.Joints[JointType.Spine].Posit
ion.Y, Tolerance));
     aktualniAnalyza.PanevRotaceSpravnost =
overPanevRotace(skeleton.Joints[JointType.HipLeft],
skeleton.Joints[JointType.HipRight],
skeleton.Joints[JointType.KneeLeft],
skeleton.Joints[JointType.KneeRight]);
     aktualniAnalyza.RamenaRotaceSpravnost =
overRamenaRotace(skeleton.Joints[JointType.ShoulderLeft],
skeleton.Joints[JointType.ShoulderRight],
skeleton.Joints[JointType.HandLeft],
skeleton.Joints[JointType.HandRight]);
```

```
     aktualniAnalyza.PohybPaziSpravnost =
overPohybPazi(skeleton.Joints[JointType.HandLeft],
skeleton.Joints[JointType.HandRight],
skeleton.Joints[JointType.FootLeft],
skeleton.Joints[JointType.FootRight]);
     aktualniAnalyza.PostaveniHlavySpravnost =
overPostaveniHlavy(headPose);

     //Nastavení pohybu hráče
     if (aktualniAnalyza.SplenoPodminekProcento >
minHranicePohybu){
          Hrac.NastavSmerPohybuHrace(zjistiSmerPohybuHrace(sk
     eleton.Joints[JointType.ShoulderCenter]));
     }
     return aktualniAnalyza;
}
```

První krok je vytvoření instance třídy `Analyza`. Postupně se její atributy plní hodnotami. Jde o jednotlivé podmínky, které musí být splněny, aby byla chůze stabilní. Pro každá z podmínek je vytvořena operace, která ji testuje a vrací hodnotu, zda je splněna nebo ne. Parametry těchto operací na ověřování podmínek jsou jednotlivé části ze skeletona, například pro ověření relaxování horních končetin, ruce musejí být volné směrem k zemi, pozice výšky ruky nemá být větší než pozice loktu atd. Po doplnění hodnot podmínek dojde ke kontrole, jestli je splněná chůze alespoň z osmdesáti procent, pokud ano, dojde nastavení směru pohybu uživatele. K tomu slouží veřejná statická metoda `NastavSmerPohybuHrace` u třídy `Hrac`. Než dojde k nastavení směru, je nutné zjistit, o jaký směr jde. Pro to se používá privátní operace `zjistiSmerPohybuHrace` s parametrem části těla zvaného `ShoulderCenter`, tedy středu ramen. Metoda je zobrazena níže.

```
private Smer zjistiSmerPohybuHrace(Joint stred){
     Smer smer = ZjistiSmer(stred.Position.Z, stredOldZ,
true);
     if (smer != Smer.Zadny){
          stredOldZ = stred.Position.Z;
          return smer;
     }else{
          smer = ZjistiSmer(stred.Position.X, stredOldX,
     false);
          if (smer != Smer.Zadny){
               stredOldX = stred.Position.X;
          }
          return smer;
     }
}
```

Cílem metody je zjistit směr pro posun uživatele. Jelikož herní plocha je tvořena souřadnicemi X a Y, je nutné provést dvě zjištění, první je pro posun dopředu a dozadu,

což odpovídá, přibližování či oddalování uživatele od zařízení kinect, bere se tedy souřadnice Z. Druhým je posun doleva nebo doprava. Opět zda se uživatel posunuje před kinectem po souřadnici X. Může nastat situace, že se uživatel nepohnul a tedy směr je žádný. Pokud ale toto nenastane, uloží se jeho aktuální souřadnice do předešlých. Jelikož zjišťování směrů nejen uživatele je použito i při ověřování podmínek, využívá tedy metodu `ZjistiSmer` s parametry určující souřadnici předešlou a aktuální, dále orientaci typu `bool`, zda nás zajímá směr dopředu, dozadu, zvolí se `true`, nebo doleva doprava, zvolí se `false`.

```
private Smer ZjistiSmer(float poziceStart, float poziceCil,
bool orientace){
     poziceStart = (float)(Math.Round(poziceStart,
PocetDesetinychMist));
     poziceCil = (float)(Math.Round(poziceCil,
PocetDesetinychMist));
     if (orientace){
          if (poziceStart < poziceCil){
               return Smer.Dopredu;
          }else if (poziceStart > poziceCil){
               return Smer.Dozadu;
          }
     }else{
          if (poziceStart < poziceCil){
               return Smer.Doleva;
          }else if (poziceStart > poziceCil){
               return Smer.Doprava;
          }
     }
     return Smer.Zadny;
}
```

Princip metody spočívá v porovnávání hodnot a dle parametru orientace vrátit směr. Jelikož jde souřadnice ze senzoru kinect jsou většinou velmi nízká desetinná čísla, je nutné provést zaokrouhlení na určitý počet desetinných míst, který je určen nastavením, zajistí nám tak potřebnou přesnost zjišťování. Pokud hodnota předešlá je větší než aktuální, znamená to, že je posun dozadu a zase naopak, nebo orientaci `false`, se řeší směru doleva, doprava. Pokud ovšem hodnoty jak předešlé tak aktuální jsou stejné, tak směr pohybu je žádný.

Pro ukázku je zde zobrazena privátní metoda ve třídě `AnalyzovatData`, pro ověření rotaci ramen. Jde o pravidlo pro stabilizaci chůze, zda rameno jde stejným směrem jako paže proti pohybu pánve. Realizace podmínek pomocí zařízení kinect není stoprocentní, proto je v aplikaci ověřováno jen šest podmínek z deseti. Složitější ověřování a práce s částmi skeletona, vytěžuje výkon počítače, proto je většina podmínek v metodách co nejjednodušší.

```
private bool overRamenaRotace(Joint shoulderLeft, Joint
shoulderRight, Joint handLeft, Joint handRight){
try{
     Smer shoulderLeftSmer =
ZjistiSmer(shoulderLeftOldPosition, shoulderLeft.Position.Z,
true);
     if (shoulderLeftSmer != Smer.Zadny){
          shoulderLeftOldPosition = shoulderLeft.Position.Z;
     }
     Smer handLeftSmer = ZjistiSmer(handLeftOldPosition,
handLeft.Position.Z, true);
     if (handLeftSmer != Smer.Zadny){
          handLeftOldPosition = handLeft.Position.Z;
     }
     //Pokud jde levé rameno jiným směrem než levá ruka
     if (shoulderLeftSmer != handLeftSmer){
          return false;
     }
     Smer shoulderRightSmer =
ZjistiSmer(shoulderRightOldPosition,
shoulderRight.Position.Z, true);
     if (shoulderRightSmer != Smer.Zadny){
          shoulderRightOldPosition =
     shoulderRight.Position.Z;
     }
     Smer handRightSmer = ZjistiSmer(handRightOldPosition,
handRight.Position.Z, true);
     if (handRightSmer != Smer.Zadny){
          handRightOldPosition = handRight.Position.Z;
     }
     //Pokud jde pravé rameno jiným směrem než pravá ruka
     if (shoulderRightSmer != handRightSmer){
          return false;
     }
     //Pokud jdou obě ramena stejným směrem.
     if (shoulderRightSmer == shoulderLeftSmer){
          return false;
     }
     return true;
}catch{
     return false;
}
}
```

Metoda `overRamenaRotace` má parametry typu Joint, jde o jednotlivé části skeletona, obě ruce a ramena. První část se zabývá zjištěním směrů levého ramena a levé ruky. Využívá k tomu již zmíněnou metodu `ZjistitSmer`. Jestliže se vydávají stejným směr, vše je v pořádku, ale jestli jde ruka jiným směrem než rameno, tak vrátí metoda `false`, jako nesplnění podmínky. To samé provede pro pravou část. Poté nastává druhá část a to

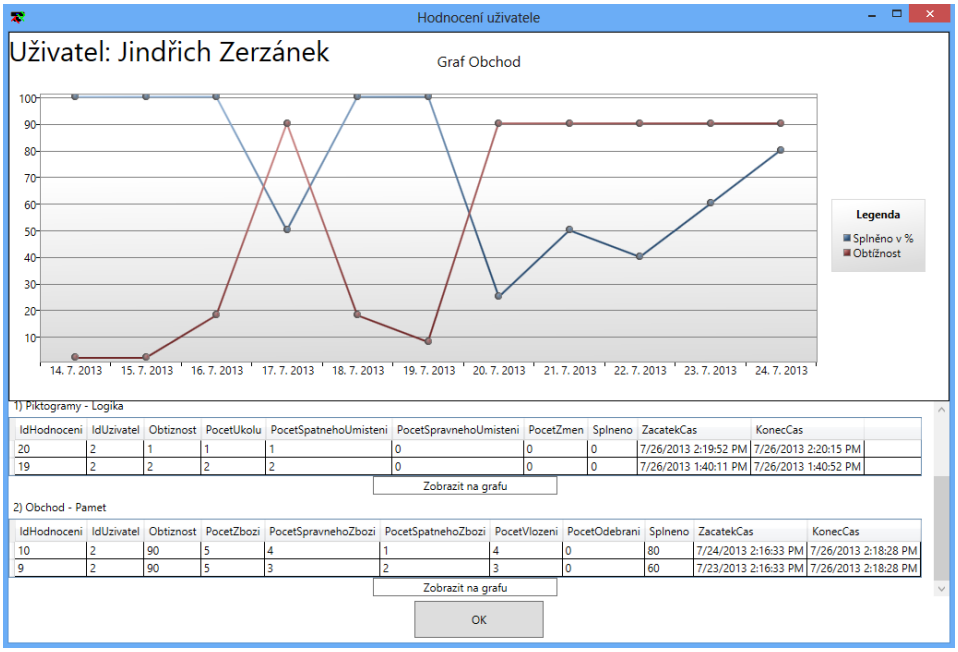
ověření, zda levé rameno a pravé mají stejný směrem, jestli ano tak opět není podmínka splněna, pokud ne tak dojde ke splnění. Obdobným způsobem se ověřuji ostatní podmínky pro stabilizaci chůze.

7 Testování systému

Tato kapitola se zabývá otestováním celkového systému, tvořeného jednotlivými aplikacemi. Při implementaci jsou důležité třídy otestovány Unit testy, jejichž cílem je ověřování správné funkčnosti dílčích částí neboli jednotek zdrojového kódu. U testovaných případů se klade důraz na závislostech na ostatních částech. Tedy při testování Unit testy jde o to, že se testovaná část izoluje od ostatních částí programu. Za tímto účelem se občas vytváří pomocné objekty, které simulují předpokládaný kontext, ve kterém testovaná část pracuje. V této práci jsou převážně otestovány třídy stereotypu control, jež řídí logiku jednotlivých aplikací. Níže je popsáno praktické otestování jednotlivých částí sytému.

7.1 Kinect Therapy Starter

Aplikace Kinect Therapy Starter je využívána terapeutem, slouží jako jednoduchá kartotéka pacientů, zobrazuje hodnocení a umožňuje spouštět vybrané moduly pro terapii. Princip použití aplikace je následující, terapeut spustí tuto centrální aplikace a v menu „Správa uživatelů“, vybere „Načíst uživatele“. Po kliknutí se zobrazí dialog, obsahující seznam s pacienty, kteří jsou v systému. Vybere si pacienta a potvrdí tlačítkem „OK“, v seznamu může být více osob a listování je pracné, proto je možné použít vyhledávání dle příjmení, stačí zadat jen prvních pár znaků a kliknout na „Hledat“. Stiskem „Ok“ dojde k načtení vybraného pacienta, veškeré informace jsou zobrazeny v hlavním formuláři. V levé části je menu s tlačítky pro jednotlivé operace, v pravé se nacházejí informace o pacientovi. Dolní část tvoří informační panel, zobrazující hlášky co se vykonalo, nebo kde nastala chyba. Operace s pacientem jsou upravení informací a odstranění. Terapeut po načtení přepíše datum poslední návštěvy, na dnešní. Poté klikne na tlačítko „Spustit“. To zobrazí dialog se seznamem správně načtených modulů. Terapeut si vybere a spustí ten, který se hodí na terapii pacienta. Pokud by nastala chyba, že se modul nespustí, je obeznámen hláškou s popisem chyby. Jakmile pacient dokončí terapii a zavře danou aplikaci, tak si terapeut zobrazí hodnocení stisknutím tlačítka „Hodnocení“. Objeví se okno, jež tvoří graf a seznam s panely. Každý panel je tvořen popiskem obsahující název modulu, tabulkou hodnocení s posledními dvěma záznamy a tlačítkem s názvem „Zobrazit na grafu“. Terapeut klikne na tlačítko u modulu, ze kterého chce zjistit, jak si pacient stojí. Vygeneruje se lineární graf zobrazující závislosti hodnoty splněno a obtížnosti hry na datu terapie. Graf obsahuje všechny pacientovy hodnoty pro vybraný modul, je z něj tedy patrné jak si stojí od začátku celé terapie. Obrázek níže ukazuje, jak takové hodnocení vypadá v této aplikaci pro Obchod modul.

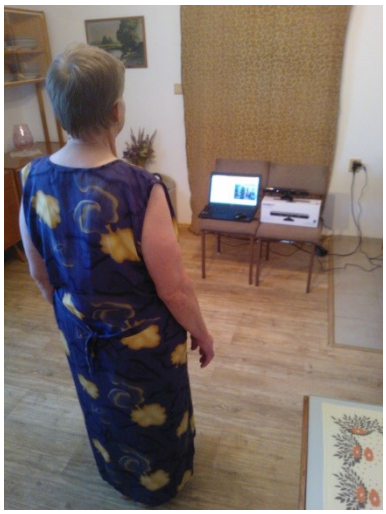


Obrázek 49 – Ukázka hodnocení pro modul Obchod. Zdroj: vlastní

7.2 Praktické testování

Pro praktické testování je zvoleno několik pacientů, důvodem je zjistit, jak se dá systém používat v reálném prostředí. Výběr pacientů se skládá ze seniora, jedná se o ženu, poté dítě předškolního věku, opět pohlaví žena, se kterými je provedeno testování jednotlivých aplikací a získání dat potřebných k vyhodnocení závěrů této práce. Test probíhá v intervalu několika dní a jsou testovány všechny tři oblasti. Níže je krátce shrnuto hodnocení pro dané osoby.

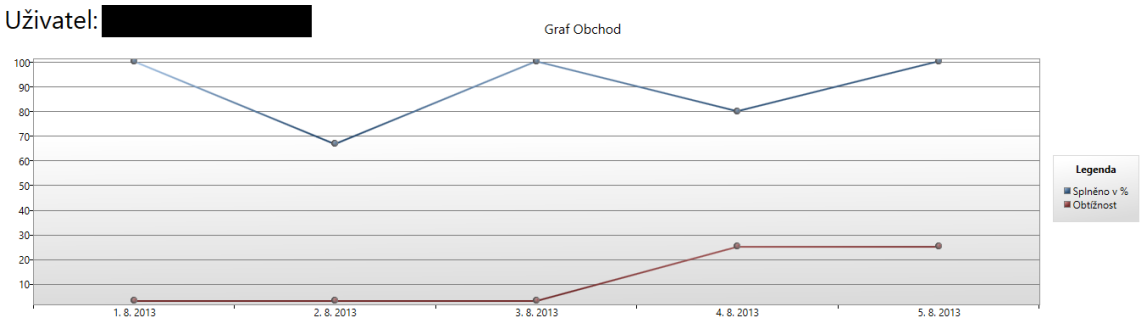
7.2.1 Senior



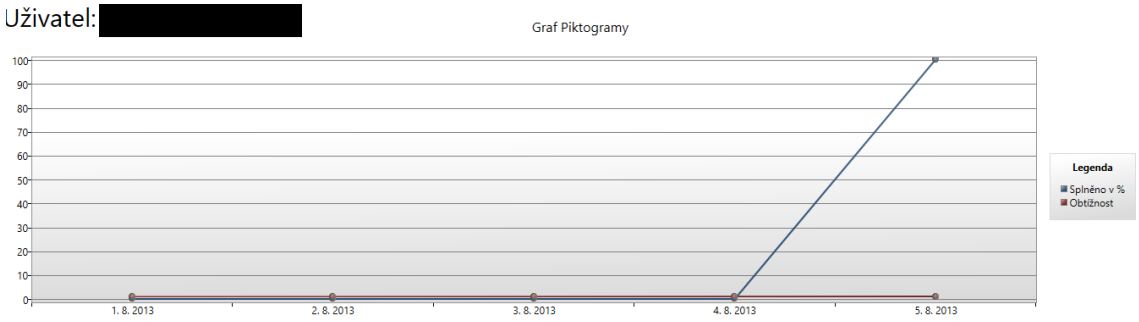
Obrázek 50 – Praktické testování systému na modulu chůze. Zdroj: vlastní

Obrázek 51 –Praktické testování systému na modulu obchod. Zdroj: vlastní

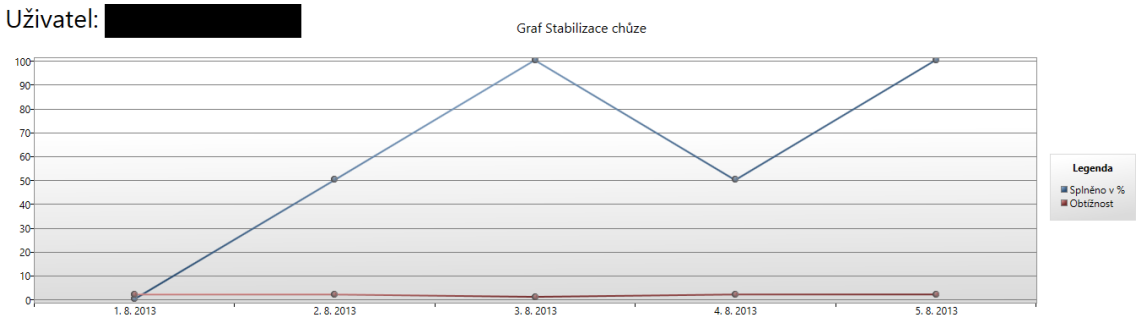
Hodnocení ženy ve věku 73let, probíhalo bez komplikací, okamžité používání ovládání senzorem Kinect bez nutnosti myši. Zde byl drobný problém při ovládání modulů Obchod a Piktogram pravou rukou, pacientce se klepala ruka a nebylo možné se zaměřit na tlačítka, proto začala používat levou ruku, u které ke klepání nedocházelo a problém byl odstraněn. Pro aplikaci Obchod modul je hodnocení následující, u úkolu se obtížnost stupňovala dvakrát, jelikož se terapie aplikovala krátce. První testování probíhalo s obtížností tři a druhé pět. Se zvyšující obtížnosti se zhoršil výsledek. U aplikace Piktogram modul byla obtížnost pouze jedna, jelikož nebyl splněn žádný úkol na sto procent z prvních čtyř měření. Při delším testování by určitě došlo k navýšení, jelikož se pacientka nesoustředila tolik na ovládání, ale více na úkol. Poslední aplikace Chůze modul, probíhala s nastavením dvou obtížností, první jedna, poté dva. Při její změně došlo k zhoršení, než si pacientka zvykla na stabilní chůzi a větší počet úkolů, na které je omezený čas. Počet sto procentní stabilizace chůze vzrůstal s každým měřením. Lze tedy říci, že docházelo ke zlepšení viditelnému v této aplikaci. Níže jsou zobrazeny grafy hodnocení pro jednotlivé moduly.



Obrázek 52 - Hodnocení seniorky z aplikace Obchod modul. Zdroj: vlastní



Obrázek 53 - Hodnocení seniorky z aplikace Piktogram modul. Zdroj: vlastní



Obrázek 54 - Hodnocení seniorky z aplikace Chůze modul. Zdroj: vlastní

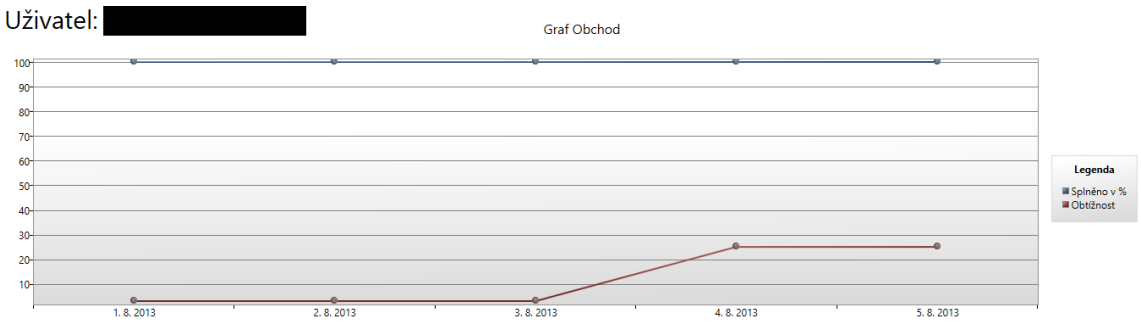
Závěr je, že během aplikování systému došlo k drobnému zlepšení schopností.

7.2.2 Dítě

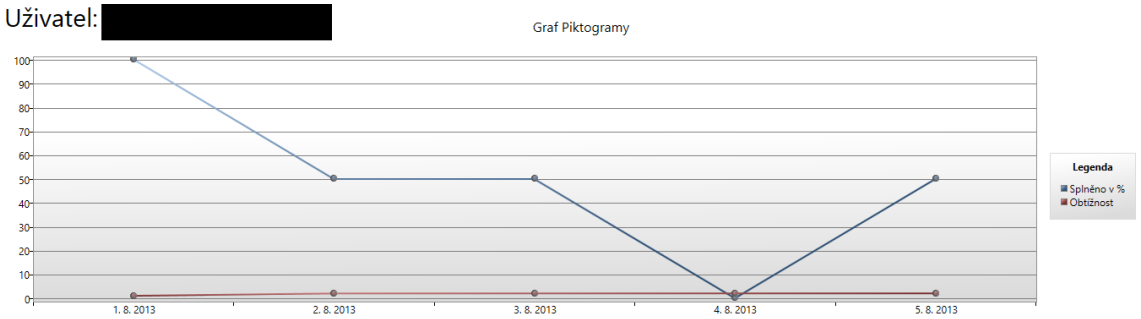
Obrázek 55 – Praktické testování systému na modulu piktogram. Zdroj: vlastní

Hodnocení dítěte ve věku 6let probíhalo v pořádku. Ovládaní senzorem Kinect bylo u aplikací Obchod a Piktogram problémové. Docházelo k problému detekci ruky, jelikož byla malá, při najetí na prvky tlačítek s posuvníkem, bylo detekováno posouvání, aniž by ruka byla v pěst. Tímto problémem se prodlužoval čas na plnění úkolu. I zde byl problém se zaměřováním na tlačítka a opět pomohlo ovládání levou rukou. Pro aplikaci Obchod modul je hodnocení následující, u úkolu se obtížnost stupňovala dvakrát, první byla tři a druhá dvacet pět. Pokaždé bylo plnění na sto procent. U aplikace Piktogram modul byly

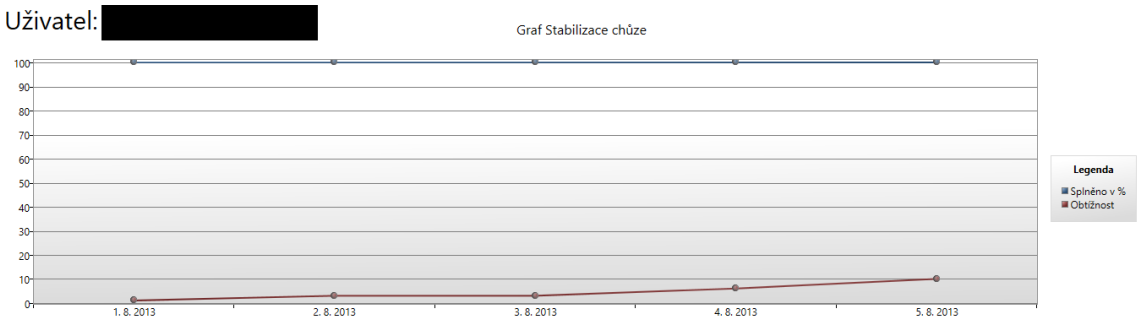
obtížnosti opět dvě, první jedna, druhá dva. Zde nastalo špatné hodnocení, kvůli jednotlivým úkolům. Byly vygenerovány složitější zadání, která pacientka bez pomoci nezvládla. Jakmile byl úkol lehčí, hodnocení se zlepšilo. Poslední aplikace Chůze modul, probíhala s postupným nastavováním obtížností jedna, tři, tři, šest, deset. Zde nebyl žádný problém a pacientka vždy splnila na sto procent i při snižování tolerance. Počet stoprocentní stabilizace chůze vzrůstala postupně každý den, poslední hodnota je čtrnáct. Níže jsou zobrazeny grafy hodnocení pro jednotlivé moduly.



Obrázek 56 - Hodnocení dítěte z aplikace Obchod modul. Zdroj: vlastní



Obrázek 57 - Hodnocení dítěte z aplikace Piktogram modul. Zdroj: vlastní



Obrázek 58 - Hodnocení dítěte z aplikace Chůze modul. Zdroj: vlastní

Závěr je, že během aplikace systému nedošlo ke zlepšení schopností.

8 Závěr

Cílem této závěrečné diplomové práce bylo navrhnout systém pro terapeutické účely využívající možnosti pohybového senzoru Kinect. Umožňuje edukativní vzdělávání pro děti předškolního věku a seniory, vyhodnocování fyzioterapických postupů, mentálního zdraví. Procvičování na souboru úloh v oblasti pohybu, logiky a paměti. Následně vyhodnotí výsledky. Vše by mělo být jednoduché a ovládání intuitivní jak pro terapeuta, tak i pacienta. Systém je navržen jako několik aplikací, jde o centrální aplikaci a moduly pro trénování jednotlivých oblastí. Pro vytvoření aplikací je využita platforma .NET a programovací jazyk C#. Pro ukládání výsledných hodnocení je použita databáze Microsoft Sql Compact Edition 4.0. Při plnění cílů nedošlo k žádným potížím a systém je plně funkční a připraven k praktickému používání. Moduly na trénování oblastí logiky, paměti a pohybu jsou tvořeny tak, aby nebyla jediná možnost ovládání zařízením Microsoft Kinect for Windows, ale i myší s klávesnicí. U systému je vhodné mít dva monitory pro přehlednost, jeden větší z nich mít pro pacienta a druhý menší zvlášť pro terapeuta. Největším problém nasazení bude vyšší cena za zařízení Microsoft Kinect for Windows a dostačující výkonný počítač.

Při tvorbě systému jsem na začátku narazil na několik problémů. Zásadním problém bylo zjištění, jaké jsou možnosti standardní terapie pro jednotlivé oblasti. Tuto problematiku jsem konzultovat s PaedDr. Zdena Šándorová, Ph.D. z fakulty zdravotnických studií Univerzity Pardubice. Po konzultacích jsme vybrali hry, které by mohli být vhodné pro dané oblasti a byla u nich možnost nasadit Microsoft Kinect for Windows. Pro oblast paměti jsme vybrali možnost trénování krátkodobé paměti, pomocí principu klasického obchodu, kde pacient dostane seznam zboží na zapamatování a poté jej musí nakoupit. Z oblasti logiky byla vybraná hra, jež využívá obrázky jako piktogramy. Cílem pacienta je dát několik těchto piktogramů do správného logického pořadí. Poslední oblastí je pohyb, zde byl velice těžký problém vymyslet nějakou úlohu využívající pohybový senzor Kinect, zvolilo se tedy trénování stabilizace chůze, jelikož je to problém začínající od dětství až po stáří. Chození stabilně předchází pádům, bolesti zad, krku, křivé páteři atd. V této aplikaci má terapeut možnost sledovat, kde přesně pacient dělá chyby. Dále pro pacienta je zde hra, která spočívá v posunování políčka reprezentujícího uživatele na jiné políčko na herní ploše, pomocí jeho chůze. Tyto tři herní aplikace jsou jen ukázkou, systém umožňuje přidávání nových her po dodržení jistých standardů.

Při implementaci bylo těžké vymyslet ovládání aplikací zařízením Microsoft Kinect for Windows a zároveň dynamické generování prvků na které senzor reaguje. Tento problém byl vyřešen použitím hotových knihoven nástroje Kinect for Windows Toolkit. Dalším složitějším problém bylo vytvoření dynamického načítání jednotlivých modulů a posléze jejich spuštění, hovoří se tedy o možnosti přidávání nových aplikací do systému. Tento problém je vyřešen rozdělením na centrální aplikace a aplikací pro terapii, centrální aplikace si načte konfigurační XML soubory modulů, které obsahují i umístění. Centrální aplikaci Kinect Therapy Starter je možné rozšířit o podrobnější informace o pacientech, složitější statistiky a lepší správy modulů. Vhodné by bylo i API rozhraní, přes které by

komunikovali jednotlivé herní aplikace. Programy pro jednotlivé oblasti terapie jsou vytvořeny jako ukázkové a jistě je lze všechny rozšířit o další možnosti, například o možnost ovládání hlasem. Podrobnější zobrazení chyb, které pacient při hraní udělal. Lepší grafická úprava.

V realizaci této diplomové práce jsem využil znalosti získané při magisterském studiu a naučil jsem se prakticky využívat technologii Kinect v aplikacích. Veškeré části systému i přes některé problémy jsou úspěšně dokončené a lze je tak využít v praktickém využití. Systém je zcela uživatelsky intuitivní a v budoucnu je možné ho rozšířit jak o nové aplikace pro terapii tak o zmíněné navrhované možnosti. Při praktickém testování se došlo k závěru, že systém přináší drobné zlepšení schopností v oblastech logiky, paměti a pohybu.

Literatura

1. **Příklady neherního využití Kinectu.** *http://wiki.knihovna.cz/.* [Online] [Citace: 17. 6 2013.] http://wiki.knihovna.cz/index.php?title=P%C5%99%C3%ADklady_nehern%C3%ADho_vyu%C5%BEit%C3%AD_Kinectu.

2. **Léčba.** *Wikipedie.* [Online] [Citace: 17. 06 2013.] http://cs.wikipedia.org/wiki/L%C3%A9%C4%8Dba.

3. **Paměť.** *Wikipedie.* [Online] [Citace: 17. 06 2013.] http://cs.wikipedia.org/wiki/Pam%C4%9B%C5%A5.

4. **Logika.** *Wikipedie.* [Online] [Citace: 17. 6 2013.] http://cs.wikipedia.org/wiki/Logika.

5. **Rytmické schopnosti tanečníků amerického stepu.** [Online] [Citace: 17. 06 2013.] http://is.muni.cz/th/136117/fsps_m/diplomka_na_tisk.txt.

6. **ŠÁNDOROVÁ, Zdeňka Ph.D.** *Základy pedagogiky a speciální pedagogiky.* Pardubice : Univerzita Pardubice Fakulkta zdravotnických studií Prezentace, 2011.

7. **Motion capture.** *Wikipedie.* [Online] [Citace: 17. 06 2013.] http://cs.wikipedia.org/wiki/Motion_capture.

8. **AB, QUALISYS.** Qualisys - The Art of Motion Capture. *CASRI.* [Online] [Citace: 24. 6 2013.] http://casri.cz/qualisys/.

9. **KinectoTherapy.** [Online] [Citace: 25. 6 2013.] http://dl.dropboxusercontent.com/u/8023254/Kinectotherapy/home.html.

10. **Serios Games.** *Wiki knihovna.* [Online] [Citace: 25. 6 2013.] http://wiki.knihovna.cz/index.php?title=Serious_game.

11. **JIANG, Kevin a MCGILL, Nicholas.** Kinect for Physical Therapy. *Kinepeutics.* [Online] [Citace: 19. 06 2013.] http://kinepeutics.blogspot.cz/.

12. **MICROSOFT.** Xbox 360 + Kinect fitness. *Xbox.* [Online] [Citace: 19. 6 2013.] http://www.xbox.com/en-US/kinect/fitness.

13. —. Develop for Kinect Microsoft Kinect for Windows. [Online] [Citace: 8. 10 2012.] http://www.microsoft.com/en-us/kinectforwindows/.

14. **Kinect for Windows Programming Guide.** *msdn.* [Online] [Citace: 2. 7 2013.] http://msdn.microsoft.com/en-us/library/hh855348.aspx.

15. **PlayStation Move.** *Wikipedia.* [Online] [Citace: 2. 7 2013.] http://en.wikipedia.org/wiki/PlayStation_Move.

16. **Wii.** *Wikipedia.* [Online] [Citace: 2. 7 2013.] http://cs.wikipedia.org/wiki/Wii.

17. **Kinect.** *Wikipedia.* [Online] [Citace: 2. 7 2013.] http://en.wikipedia.org/wiki/Kinect.

18. **KUBICA, PETER.** Uživatelské rozhraní založené na zpracování hloubkové mapy. [Online] Diplomová práce. FIT VUT Brno. [Citace: 2. 7 2013.] http://www.fit.vutbr.cz/study/DP/all.php?id=15610&file=t.

19. **Pro koordinaci a stabilizaci chůze.** *SM System.* [Online] [Citace: 22. 7 2013.] http://www.smsystem.cz/index.php?option=com_content&view=article&id=135&Itemid=104&lang=cs.

20. **Enterprise Architect.** *Wikipedie.* [Online] [Citace: 22. 7 2013.] http://cs.wikipedia.org/wiki/Enterprise_Architect.

21. **Meca, Jan.** *Unified Modeling Language (UML).* [Prezentace] Bruntál : Střední průmyslová škola Bruntál, 2007.

22. **Návrhové vzory.** *Wikipedie.* [Online] [Citace: 30. 7 2013.] http://cs.wikipedia.org/wiki/N%C3%A1vrhov%C3%BD_vzor.

23. **Singleton.** *Wikipedie.* [Online] [Citace: 30. 7 2013.] http://cs.wikipedia.org/wiki/Singleton.

24. **WebML - datové modelování.** *Interval.cz.* [Online] [Citace: 30. 7 2013.] http://interval.cz/clanky/webml-datove-modelovani/.

25. **Implementace.** *Wikipedie.* [Online] [Citace: 31. 7 2013.] http://cs.wikipedia.org/wiki/Implementace.

26. **Microsoft Visual Studio.** *Microsoft.* [Online] Microsoft. [Citace: 31. 7 2013.] http://www.microsoft.com/cze/msdn/vstudio/.

27. **C sharp.** *Wikipedie.* [Online] [Citace: 31. 7 2013.] http://cs.wikipedia.org/wiki/C_Sharp.

28. **Windows presentation foundation.** *Wikipedie.* [Online] [Citace: 31. 7 2013.] http://cs.wikipedia.org/wiki/Windows_Presentation_Foundation.

29. **Microsoft SQL Server Compact 4.0.** *Microsoft.* [Online] Microsoft. [Citace: 31. 7 2013.] http://www.microsoft.com/cs-cz/download/details.aspx?id=17876.

30. **MICROSOFT.** Knihovna MSDN. [Online] [Citace: 8. 10 2012.] http://msdn.microsoft.com/cs-cz/library/default.aspx.

31. **JETENSKÝ, Pavel.** Use your desk as mouse-free multitouch interface with MS Kinect. [Online] 8. 10 2012. http://jetensky.net/blog/2011/12/18/touchtable/#more-140.

32. **EVJEN, Bill, a další.** *C# 2008 Programujeme profesionálně.* místo neznámé : COMPUTER PRESS, 2009. EAN 9788025124017.

33. **CZOMPÁL, Zsolt.** Ovládání desktopu pomocí senzoru Kinect. [Online] Diplomová práce. FIT VUT Brno. [Citace: 8. 10 2012.] http://www.fit.vutbr.cz/study/DP/DP.php?id=11332&y=2011&k=Kinect.

34. **CZOMPÁL Zsolt, Ing.** Ovládání desktopu pomocí senzoru Kinect. [Online] Diplomová práce. FIT VUT Brno. [Citace: 8. 10 2012.] http://www.fit.vutbr.cz/study/DP/DP.php?id=11332&y=2011&k=Kinect.