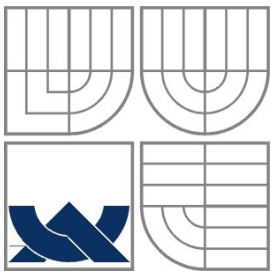
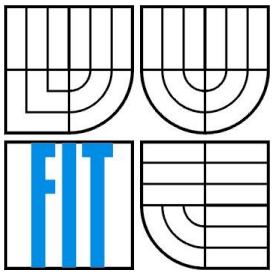


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

SYSTÉM PRO SPRÁVU A SIMULACI STAVU VČELSTEV

SIMULATION AND MANAGEMENT SYSTEM FOR BEEKEEPERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK KAŇOVSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Dr. Ing. PETR PERINGER

BRNO 2013

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem a implementací systému pro evidenci a simulaci stavu včelstev. Evidenční systém umožňuje ukládání, aktualizaci a prohlížení stavu včelstev a je navržen především pro nástavkové typy úlů. Simulační model je zaměřen na simulaci medové snůšky a s evidenčním systémem je do jisté míry provázán. V závislosti na nastavených parametrech simulačního modelu jsou při simulaci zobrazována místa, kam by včely pravděpodobně létaly za snůškou, a celkový výnos medu za jeden den.

Abstract

This bachelor thesis deals with analysis, design and implementation of the management and simulation system for beekeepers. The management system provides storing, updating and browsing the state of bee hives and is designed for the type of hives with removable frames. The simulation system contains the model of collecting nectar by honey bees and is addapted to management system. Depending on parameters of simulation model the application visualizes locations where would honey bees probably fly for the nectar and shows the amount of collected honey in one day.

Klíčová slova

včely, nástavkové včelaření, evidence včelstev, simulace včelstev, model medové snůšky

Keywords

bees, beekeeping, system for beekeepers, honey bee simulator, collecting nectar model

Citace

KAŇOVSKÝ, Zdeněk. *Systém pro správu a simulaci stavu včelstev*. Brno, 2013. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

Systém pro správu a simulaci stavu včelstev

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Dr. Ing. Petra Peringera. Další informace mi poskytli Ing. Martin Krsek, CSc., a Ing. Antonín Přidal, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Zdeněk Kaňovský
15. květen 2013

Poděkování

Děkuji panu Dr. Ing. Petru Peringerovi za pomoc při zpracování bakalářské práce, Ing. Martinu Krskovi, CSc., a Ing. Antonínu Přidalovi, Ph.D., za poskytnutí informací týkající se vlastností včel.

Obsah

1 Úvod

Včelařství je jedním z nejstarších oborů lidské činnosti. Postupem času s vývojem nových technologií vznikaly nové způsoby včelaření. Stejně jako v jiných oblastech lidské činnosti, většina nových technologií se snaží o co nejsnazší manipulaci a největší přínos. Pokud to lze, je výhodné pro vybrané situace vytvořit simulační modely a jejich simulací zjistit, za jakých podmínek bude daná činnost nejvýhodnější. V oboru včelařství se může například jednat o rozmístění včelstev v okolí za účelem co nejefektivnějšího opylení plodin a nejvyšší medové snůšky.

Pro simulaci je nutné znát jisté parametry, aby dosahovala určité věrohodnosti. Vzniká tak potřeba evidovat informace o včelstvech a jejich okolí. K tomu může být použit evidenční systém na simulačním modelu zcela nezávislý. Evidenční systém nemusí sloužit jen jako zdroj vstupních dat pro simulační modely, ale může včelaři poskytovat přehledy o včelstvech a jejich stavech.

Cílem této práce je vytvořit informační systém pro podporu chovu včel. Systém je rozdělen na 2 části: první část systému, umožňující vkládat a prohlížet informace o včelstvech, a druhou část, umožňující předpověď vývoje včelstev v oblasti medové snůšky. Aplikace může sloužit včelařům pro přehled včelstev a plánování dalších úkonů.

Obsahem následujícího textu je stručný úvod do problematiky včelařství, popis návrhu evidenčního systému včelstev a simulátoru medové snůšky, jejich implementace a testování aplikace. V kapitole Přehled současného stavu je popsána analýza současného stavu včelařství v České republice. Jsou zde rozebrány některé problematiky související s chovem včelstev. Dále jsou zde popsány důvody vedení evidence včelstev, význam počítačové simulace a uvedeny některé existující simulační modely. Z důvodů své zajímavosti a přínosnosti je pro simulaci zvolen model medové snůšky.

Kapitola Návrh aplikace se zabývá objektově orientovaným návrhem aplikace tvořené systémem pro evidenci stavu včelstev a simulačním modelem medové snůšky. Obsahuje identifikaci důležitých objektů a jejich rozdělení do příslušných tříd. Návrh systému pro evidenci včelstev byl přizpůsoben rozmáhajícím se moderním trendům, konkrétně nástavkovému stylu včelaření, s cíli jednoduchého a rychlého ovládání a co nejmenších nároků na uživatele. Tento systém nebyl inspirován žádnými evidenčními systémy stejného nebo podobného zaměření. Návrh simulačního modelu medové snůšky respektuje reálný model, vyjma hledání a vybírání nejvhodnější plodiny včelami průzkumnicemi. V tomto modelu jsou lokace nejvýhodnějších plodin předem známy.

Kapitola Implementace programu popisuje implementaci programu ve vývojovém prostředí Qt. Jsou v ní uvedeny diagramy tříd evidenční i simulační části aplikace, které odpovídají jejich skutečné implementaci. V předposlední kapitole Testování programu je kromě testování aplikace diskutována validita vytvořeného simulační modelu

2 Přehled současného stavu

V následujícím textu jsou použity termíny související se včelařstvím a počítačovou simulací. Vysvětlení všech těchto termínů je nad rámec této práce. Pro jejich pochopení je vhodné nastudování příslušné problematiky z jiných zdrojů – například [13] a [14].

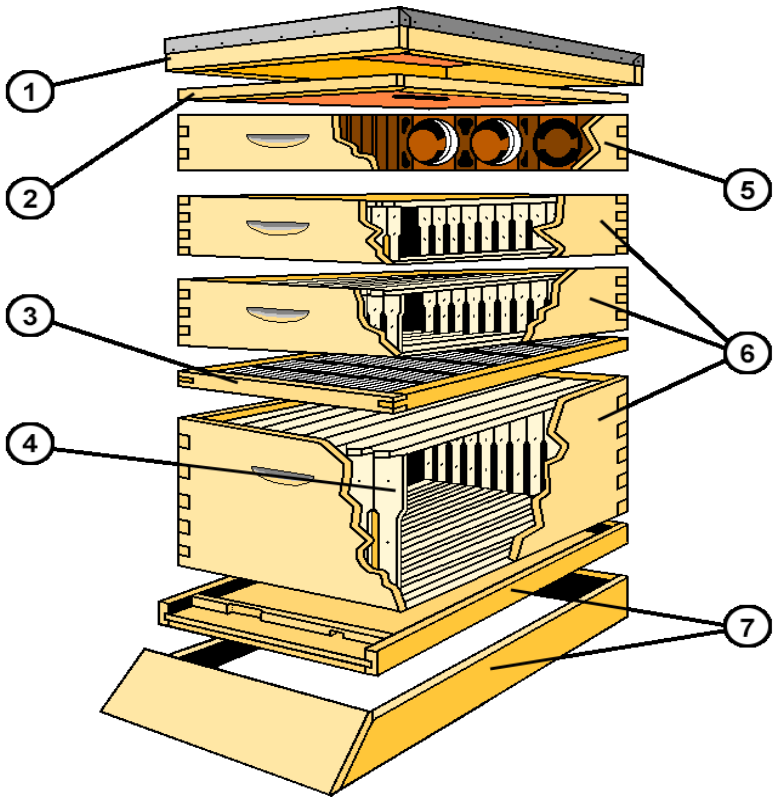
Zajímavou a v některých oblastech nepříliš probádanou problematikou jsou společenství včel a jejich chování. Mnoho včelařů je schopno pracovat se včelstvy a mít z nich užitek, přesto příčiny jejich chování neznají. Stejně tak i vědci zkoumající chování včel dodnes některé jejich chování a jeho příčiny nedokáží vysvětlit.

Je tedy složité vytvořit věrohodný simulační model související s činností včelstev. Jednoduší je vytvořit informační systém zaznamenávající události vznikající při chovu včelstev, který na základě vložených údajů může pomoci chování včelstev pochopit.

2.1 Problematika chovu včelstev

Mezi časté činnosti včelstev patří například sběr pylu a nektaru, přeměna nektaru na med, jeho ukládání do pláství, interakce včel s ostatními včelami, atd. Problémy, s nimiž se včelstvo může potýkat, mohou být nemoci, vykrádání zásob cizími včelami nebo jiným hmyzem, velmi nízké okolní teploty a další. V následujících podkapitolách jsou blíže popsány některé z nich. Pro lepší představu je níže uvedeno schéma nástavkového včelího úlu.

Obr. 2.1 Schéma nástavkového úlu [22]: **1** – úlové víko, **2** – poklop nebo strupková fólie, **3** – mateří mřížka, **4** – rámky, **5** – krmítko, **6** – nástavky, **7** – úlové dno



2.1.1 Mor včelího plodu

Vážnou hrozbou pro včelstva je v dnešní době mor včelího plodu [8]. Larvy včel zasažené touto chorobou hynou ještě před vylíhnutím, a tak populace včelstva postupně klesá, až nakonec zaniká celé včelstvo. Takto zasažené úly včetně všeho nářadí a pomůcek z hořlavého materiálu, které se včelstvem přišly do styku, se musí tepelně zlikvidovat. Včelař tak může přijít i o všechna svá včelstva.

Tomu by šlo předejít, pokud by byl k dispozici model šíření této nemoci a aktuální informace o okolních nakažených včelstvech. Tento model by mohl být založen na již známých modelech šíření nemocí a dosahovat tak už zpočátku určité úrovně věrohodnosti.

Obr. 2.2 a 2.3 Mor včelího plodu [23], [24]

2.1.2 Medová snůška

Hlavním důvodem, proč je většina včelstev chována, je zisk medu. Ten vzniká přeměnou nektaru z květů rostlin. Nektar je získáván následovně. Z úlu vylétávají včely průzkumnice do všech směrů, přičemž letový rádius včel činí vzdálenost 3 až 5 km [1][2]. Pokud včela narazí na ceněnou plodinu, obsahující množství nektaru nebo pylu, zapamatuje si její polohu a vrací se do úlu. Tam ostatním včelám pomocí „včelích tanečků“ předá informaci o nalezené plodině a její poloze. Včely létavky pak na místo odlétají a nektar sbírají [9].

Okolností majících vliv na medovou snůšku je mnoho. Některé z nich jsou:

- počasí, teplota a vlhkost vzduchu,
- druh a množství včel,
- postupy a umění včelaře,
- okolní prostředí – nadmořská výška, znečištění prostředí,
- množství, rozmístění a stadium květenství okolních plodin.

Silné včelstvo může mít v letním období kolem 60 000 včel [4][5], z nichž včely létavky, sbírající nektar, tvoří asi jednu třetinu [1]. Letová rychlost včely se pohybuje od 6 do 8 m/s, tj. přibližně 21 až 29 km/h, při zatížení rychlost klesá asi na polovinu [1][2]. Při jednom výletu včela

navštíví průměrně 80 květů [2][3] a může nasbírat kolem 40 až 50 mg nektaru obsahujícího 50% vody [4]. Nektar je po návratu včely do úlu 15 až 20 minut předáván ostatním včelám, které jej míchají s výměšky svých hltanových žláz a vzniklou tekutinu ukládají do pláství [1]. Z původního množství nasbíraného nektaru pak vznikne přibližně 30 mg medu s obsahem 18 % vody [4].

Obr. 2.4 Pozorované parametry včel při sběru nektaru [10]

Tab. 3 – Komonice, 10.20 hod., 30. 6. 1986, Liptovský Hrádok, včelstvo C1

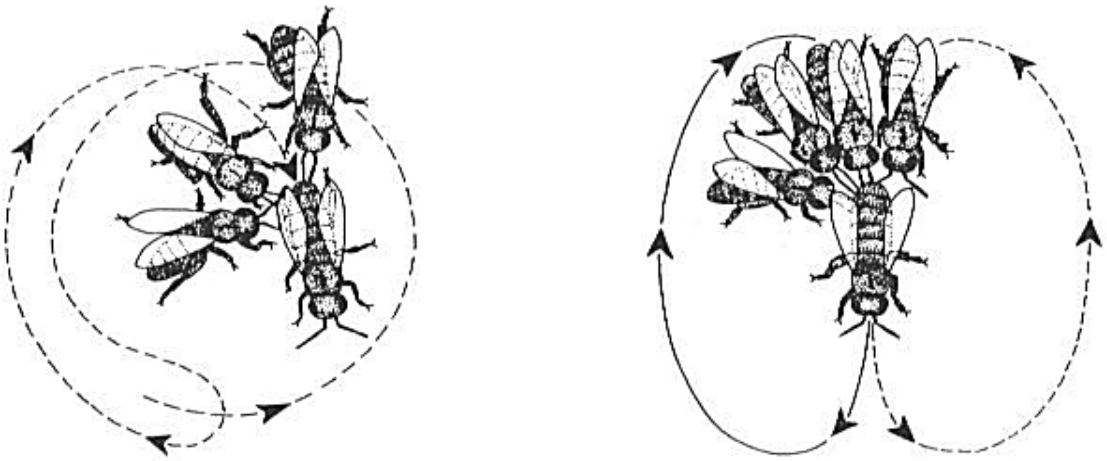
Včela	c (%)	V (µl)	S (mg)	m_2 (mg)	δ (mg)	e_L (J)	e_S (J)	e_N (J)	e_N/e_S
1	29,0	30,2	9,8	116,9	36,9	219,5	44,6	174,9	3,9
2	21,5	44,5	10,4	131,8	51,8	229,6	48,1	181,5	3,8
3	26,5	33,6	9,9	120,4	40,4	220,3	45,4	174,8	3,8
4	22,3	42,0	10,2	129,2	49,2	226,2	47,5	178,7	3,8
5	19,0	31,9	6,5	117,8	37,8	160,6	44,8	115,8	2,6
6	19,0	30,2	6,2	115,9	35,9	154,3	44,4	110,0	2,5
7	21,0	56,3	12,8	144,7	64,7	272,9	51,1	221,8	4,3
8	27,5	35,3	10,8	122,5	42,5	236,8	45,9	190,9	4,2
9	13,5	32,8	4,7	118,4	38,4	127,6	45,0	82,6	1,8
10	42,0	30,2	15,0	118,8	38,8	312,9	45,1	267,8	5,9
11	24,5	49,6	13,4	138,0	58,0	281,9	49,5	232,4	4,7
12	31,5	38,6	13,8	126,8	46,8	288,9	46,9	242,0	5,2
13	19,0	42,8	8,8	129,6	49,6	200,6	47,6	153,0	3,2
14	28,5	37,0	11,8	124,5	44,5	254,1	46,4	207,7	4,5
15	41,5	38,6	19,0	128,7	48,7	380,3	47,4	332,9	7,0
16	17,3	16,8	3,1	101,3	21,3	99,0	41,0	58,1	1,4
17	22,0	45,5	10,9	133,0	53,0	238,5	48,4	190,1	3,9
18	17,5	37,8	7,1	124,0	44,0	170,7	46,3	124,4	2,7
Průměr	**24,6**	**37,4**	**10,2**	**124,6**	**44,6**	**226,4**	**46,4**	**180,0**	**3,8**

Vzdálenost úlu od porostu = 400 m, doba pobytu létavky mimo úl = 25 minut, hmotnost vyletující včely = 80 mg, podíl času sběru sladiny včelou : času strávenému v porostu = 50 %

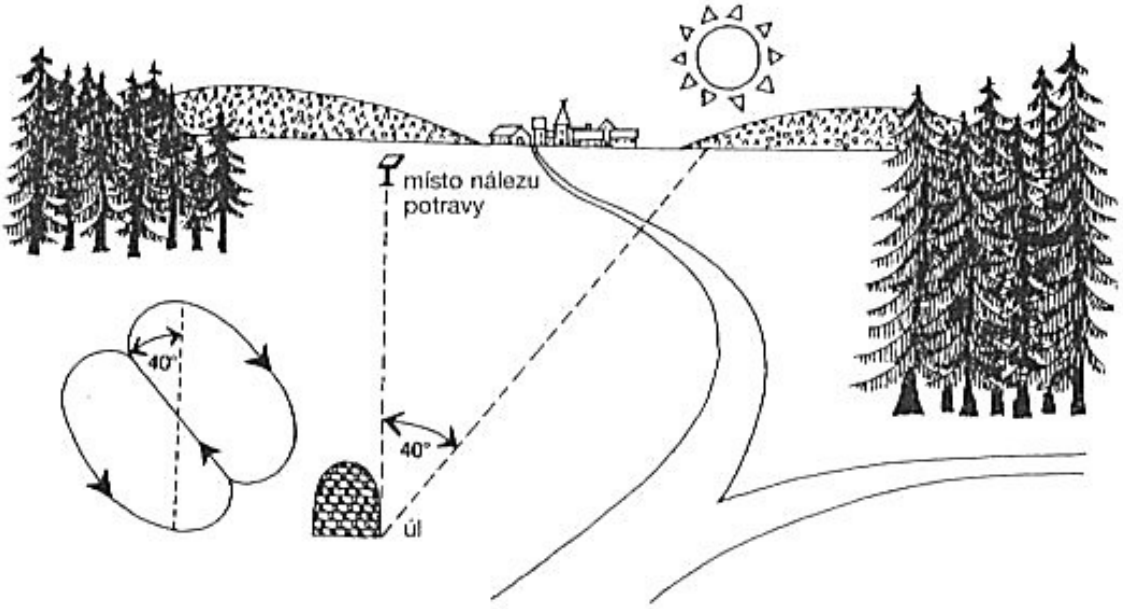
Význam symbolů v tabulkách:
c – koncentrace včelou přinášené sladiny, %
V – objem přinášené sladiny, µl
S – obsah sušiny (cukrů) ve sladině, mg
m_2 – hmotnost včely při skončení sběru sladiny, před návratem do úlu, mg
δ – přírůstek hmotnosti včely (nákladu sladiny) za dobu sběru, mg
e_L – včelou nasbíraná energie v porostu, J
e_S – spotřeba energie včelou při letu do porostu + při sběru sladiny + při návratu do úlu, J
e_N – čistý zisk energie (netto) během jednoho výletu pro potravu, J
e_N/e_S – poměr čistého zisku energie ku výdajům energie na celý let včely; efektivnost práce včely

Z výše uvedených hodnot lze odvodit čas včelou strávený přímým sběrem nektaru na asi 13 minut. Při počtu 80 navštívených květů se včela na jednom květu zdrží průměrně 10 sekund. Nejvíce včely létají za slunného počasí. Při zvýšené oblačnosti či občasných přeháňkách letová aktivita včel pozvolna klesá, v plném dešti již nelétají vůbec. Co se týče teploty, letová aktivita včel začíná při 6 - 10°C, plnou pohyblivost dosahuje včela při 15°C [1]. Včely z úlu začínají vylétat při východu slunce a létají až do soumraku.

Obr. 2.5 Ukázka včelích tanečků [25]: *vlevo* - kruhový taneček, oznámení zdroje snůšky od 50 do 100 m od úlu, *vpravo* - natřásavý (osmičkový) taneček, zdroj snůšky je vzdálen více než 100 metrů [1].



Obr. 2.6 Nákres orientace včel v terénu [25]: úhel, který udává včela, je dán polohou slunce ke svislici.



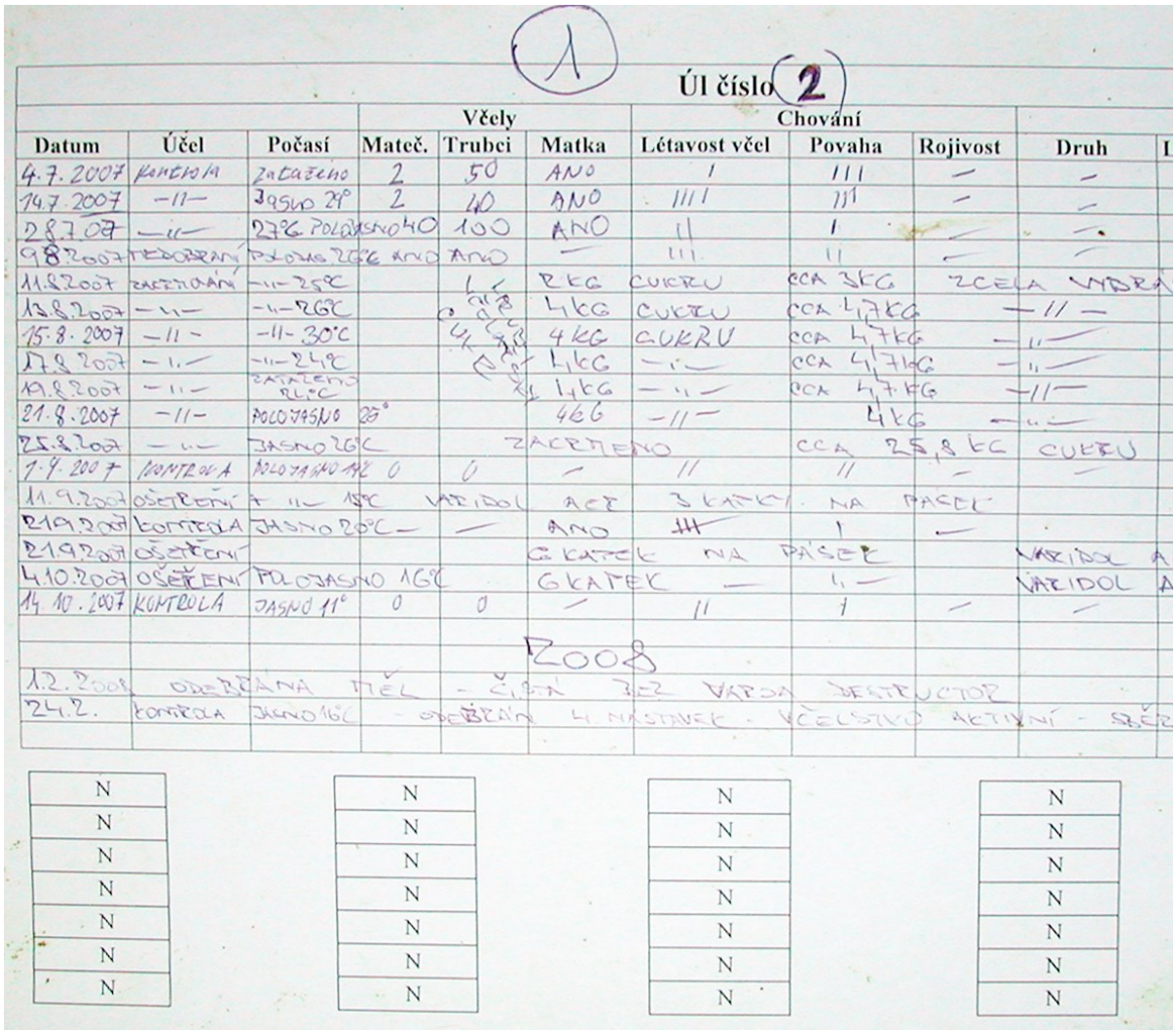
Jelikož se hodnoty jednotlivých parametrů mající vliv na medovou snůšku v různých zdrojích liší, a to více či méně, je pro výpočet simulace zvolen buď průměr těchto hodnot, nebo hodnota uváděná více zdroji.

2.2 Vedení evidence o stavu včelstev

Evidence včelstev představuje vedení informací o včelstvech a událostech s nimi spojenými. Téměř každý včelař si určitým způsobem eviduje stav svých včelstev. To je důležité nejen kvůli přehledu, ale i kvůli plánování budoucích zásahů do včelstev – kontrolní prohlídky, přidávání či odebírání nástavků, výměna rámků, vytváření oddělků, atd. Bez informací o aktuálním stavu včelstev je plánování těchto akcí složitější, méně efektivní a často i ztrátové.

Evidence stavu včelstev může být vedena jako každá jiná evidence podobného typu – písemně či elektronicky. Hlavní nevýhody **písemné evidence** spočívají v nemožnosti automaticky agregovat jisté veličiny - například celkový výnos medu za daný rok, průměrné množství spotřebovaného krmiva, atd. Další nevýhodou je pomalejší procházení záznamů. Mezi výhody patří dostupnost takové evidence, možnost psát údaje již během prohlídek včelstev a téměř nulové náklady na pořízení a vedení evidence.

Obr. 2.7 Vedení písemné evidence o provedených prohlídkách včelstev



Elektronická evidence může poskytovat řadu automatizovaných funkcí pro zjednodušení a zrychlení práce s evidencí:

- agregační funkce – např. průměrný počet nástavků,
- rychlé vkládání – minimální režie uživatele,
- grafické zobrazení – převedení zadaných textových údajů do grafické podoby, čímž se zvýší přehlednost celého systému,
- animace provedených změn – např. vývoj úlů od začátku roku.

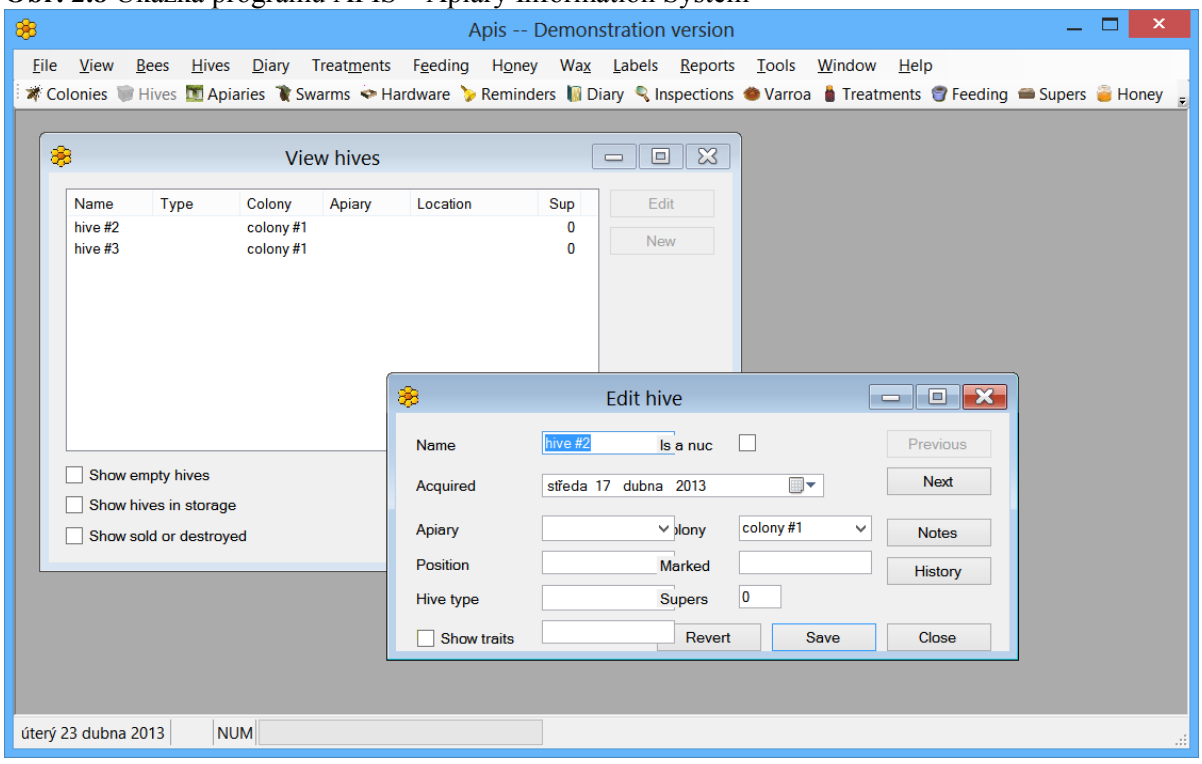
Její výhody se nejvíce projeví s přibývajícím počtem evidovaných včelstev. Nejméně náročný na prostředky se jeví způsob vedení evidence v tabulkovém procesoru, kdy si včelař

individuálně stanoví, které informace bude vést a které nikoliv. Specializované programy jsou naproti tomu specifičtější a poskytují vlastní množinu vedených informací. To však nemusí včelaři vždy vyhovovat.

2.3 Existující program pro evidenci stavu včelstev

Jedním z dostupných programů pro evidenci včelstev je **APIS – Apiary information system** [6]. Program umožňuje evidovat mnoho detailů o včelstvech, jejich prohlídkách, nemocech, léčení, zakrmování, nakupování a prodávání, rozlišovat včelaře, vkládat upomínky do kalendáře a další. Je v angličtině a má pouze textové tabulkové rozhraní. Vhodnost programu k přehledné a rychlé evidenci včelstev tak záleží na individuálním posouzení. Na webu produktu je nabízena zkušební 30denní verze tohoto programu. Po uplynutí této doby je třeba vložit licenční klíč, jehož cena činila ke dni 1. 5. 2013 £70.

Obr. 2.8 Ukázka programu APIS – Apiary Information System



2.4 Simulační modely včelstev

Existuje řada modelů zabývajících se aktivitami včelstev. Některé se zaměřují na modelování problematik spjatých s životem včelích kolonií, jako jsou např. dynamický růst včelstev v průběhu roku, měnící se množství roztočů ve včelstvu či hledání medové snůšky. Jiné modely jsou vytvořeny za účelem optimalizace již existujících algoritmů používaných v informačních technologiích ve snaze napodobit jisté postupy včel, utvářené po miliony let.

2.4.1 Existující simulační modely

BEEPOP: A honeybee population dynamics simulation [15] je simulační model zaměřený na dynamický růst včelstev v průběhu roku, závislý na počátečním stavu včelstva, počasí a teplotě, síle a potenciálu včelí královny klást vajíčka, době života včel a dalších. Výsledky simulace tohoto modelu naznačují, že největší vliv na sílu včelstva má včelí matka a její potenciál v kladení vajíček.

A population model for the ectoparasitic mite Varroa jacobsoni in honey bee (Apis mellifera) colonies [16] se zabývá vlivem roztočů Varroa jacobsoni na zkrácení doby života včel. Používá model simulující dynamickou velikost populace roztočů, která se může měnit každým dnem. Umožňuje předpověď úmrtnosti včel na základě prezence či absence včelího plodu, což má včelaři pomoci zlepšit způsob včelaření.

Application of honey-bee mating optimization algorithm on clustering [17] navrhuje optimální algoritmus třídící data do shluků na základě jejich vlastností. Inspirace pochází z páření včelí matky. Vlastní algoritmus je srovnáván se známým algoritmem K-means [18].

A model of collective nectar source selection by honey bees: Self-organization through simple rules [19] modeluje výběr nejvhodnějšího zdroje nektaru v okolí včelstva. Zohledňuje i zdroje, kde včely nektar sbíraly dříve. Použité algoritmy mají předlohu ve skutečném chování včel. Tento model není veřejně zdarma dostupný, nejsou tedy známy ani implementace použitých algoritmů.

2.4.2 Výběr simulačního modelu

Každý z výše uvedených modelů je ve své kategorii jistě přínosný. Z pohledu včelařů by zřejmě byla nejprospěšnější předpověď medové snůšky. Přestože většina včelařů v ČR má své úly rozmístěny staticky, někteří se svými úly kočují, tj. dynamicky je přemísťují. Bylo by pro ně přínosné znát nejvýhodnější polohu umístění úlů. Lepší poloha znamená rychlejší opylení plodiny a větší výnosy medu. Důležité, jak pro sadaře, tak pro včelaře, je i rozmístění optimálního počtu včelstev na danou plochu. Z těchto důvodů se zdá být model medové snůšky pro aplikaci velmi zajímavým.

3 Návrh aplikace

V počátku návrhu byla vyvíjená aplikace pojmenována **APIS - Apiarist Information System.** Později byla zjištěna podoba tohoto názvu s již zmíněným existujícím programem APIS – Apiary Information System [6]. Shoda názvu je ze strany vyvíjeného programu čistě náhodná, název vznikl spojením slov Api a IS v APIS.

Aplikace byla rozdělena na 2 části: evidenci stavu včelstev a simulaci medové snůšky. Evidence stavu včelstev umožňuje vkládat nové informace o včelstvech a zobrazovat jejich historii. Simulace medové snůšky umožňuje předpověď lokací, kam budou pravděpodobně včely létat za snůškou, a množství nasbíraného medu. Obě části tvoří samostatné systémy, přičemž simulace medové snůšky je do jisté míry závislá na evidenčním systému kvůli získání informací o stavu včelstev.

Cílová skupina uživatelů jsou včelaři s nástavkovými úly. S přihlédnutím k průměrnému věku včelařů a jejich schopnostem by mělo být rozhraní tohoto programu navrženo jednoduše, přehledně, s intuitivním ovládáním. Především pro včelaře s více včelstvy je téměř nezbytné, aby zadávání prohlídek do evidenčního systému trvalo co nejkratší čas a zbytečně nezdržovalo včelaře zadáváním irelevantních údajů.

V následujícím textu je popisován návrh programu v paradigmatu objektově orientovaného programování - OOP. Z důvodu omezeného rozsahu práce není její součástí úvod do problematiky objektově orientovaného programování. Pro pochopení návrhu programu je nutné znát alespoň základy tohoto paradigmatu. Základní přehled problematiky je popsán například v dokumentu Úvod do OOP [20] dostupného elektronicky online.

3.1 Návrh evidence stavu včelstev

Existují různé typy včelaření s různými typy úlů. Metodiky se mohou již v základu lišit, což znamená nutnost přizpůsobení evidenčního systému těmto okolnostem. Mezi moderní typy včelaření patří nástavkové včelaření. Hlavními rysy, oproti klasickému včelaření, je manipulace s nástavky - přidávání, odebírání či přesouvání nástavků. Evidenční systém by měl tyto funkce podporovat. Množina operací prováděná s jiným než nástavkovým úlem je ve velké většině podmnožinou operací s nástavkovým úlem. Informační systém primárně určený pro nástavkové úly by tedy měl být bez větších překážek použitelný i pro evidenci úlů jiných než nástavkových.

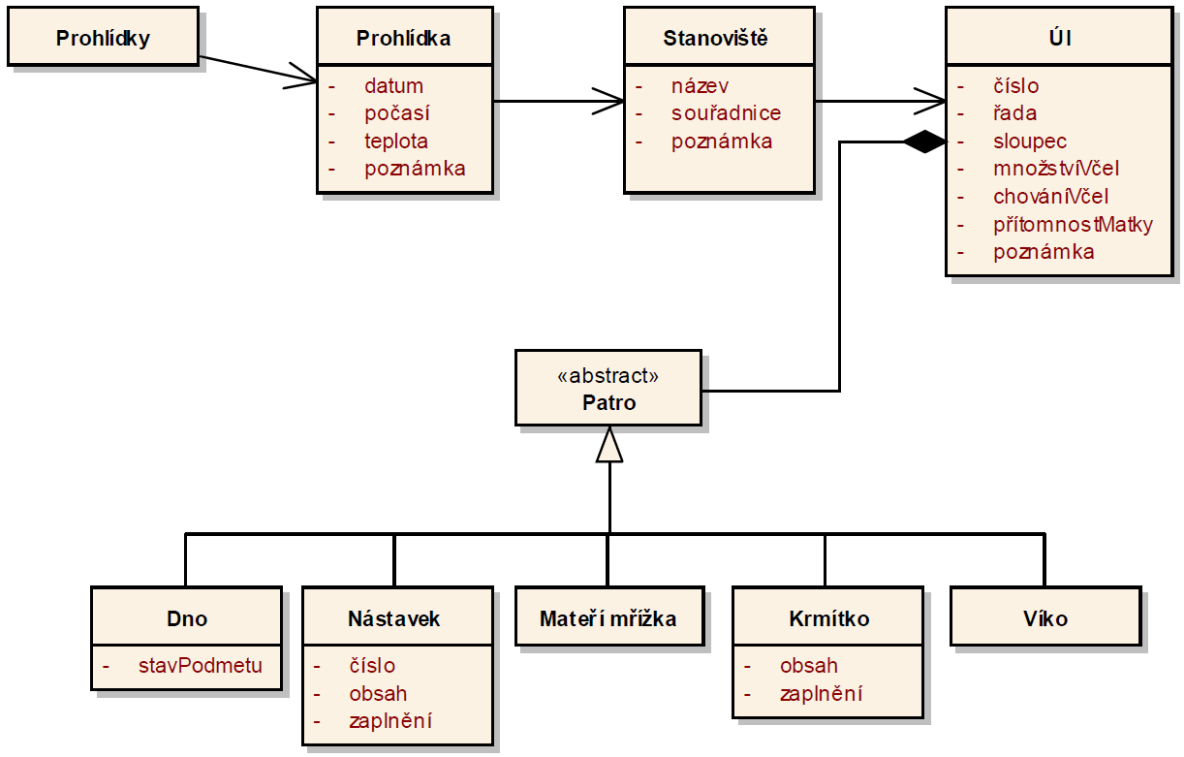
3.1.1 Množství detailů evidence

Jako každá obecná evidence i evidence včelstev by měla poskytovat variabilní množství vedených údajů. Nevhodná praktika, používaná v některých programech, je vyžadování často nepotřebných informací. Pokud uživatel bude chtít vést pouze přehled úlů, není vhodné jej nutit vyplnit informace o číslování úlů, druhu a množství včel, přítomnosti včelí matky, atd. Naopak program musí umožňovat evidenci detailnějších informací o úlu, například chování včel, přítomnost matky, v jakém nástavku se nachází plod a jeho množství a další, a informací o stanovišti – na jakých souřadnicích se nachází, kolik obsahuje včelstev či kdy byla provedena poslední prohlídka.

3.1.2 Diagram tříd evidenčního systému

Abstrakcí jistých skutečností z reálného světa lze získat několik základních objektů reprezentujících sféru včelaření. Objekty jsou rozděleny tak, aby každý představoval samostatnou část problematiky a bylo možné jej nezávisle na ostatních objektech měnit. Stejné typy objektů mají společnou šablonu – třídu.

Obr. 3.1 Diagram tříd evidenčního systému



Třída Prohlídka

Evidovaný stav včelstev se obvykle mění při každé prohlídce. Prohlídkou je myšlena kontrola včelstev nebo manipulace s nimi. Prohlídka se zpravidla provádí po 1 až 2 týdnech, jako její identifikátor může být použito příslušné datum. Je vhodné evidovat informace o počasí, teplotě a umožnit uživateli vložení individuální poznámky. Instance třídy `Prohlídka` obsahují další objekty a podobjekty, které tvoří její konfiguraci – stav včelstev k danému datu.

Třída Stanoviště

Entita zastupující skupinu úlu se nazývá stanoviště. Seskupuje úly umístěné v jedné lokaci. Identifikátorem konkrétního stanoviště je jeho název. Někteří včelaři věnující se kočování přesouvají najednou celé stanoviště, neboli kočovné zařízení, na jinou lokaci. Především u takového typu stanoviště je vhodné evidovat informace o jeho umístění.

Třída Úl

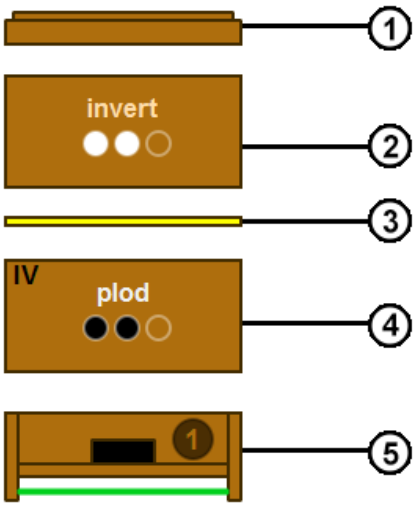
Včelstva jsou umístěna v objektech zvaných úly. Úly mohou být různých druhů a typů, pro zjednodušení byl vybrán obecný nástavkový úl. Instance třídy `Úl` by měly obsahovat informace o

včelstvu – síle včelstva a přítomnosti včelí matky. Úl přestavuje včelí obydlí, jež bývá často identifikováno svým číslem. Fyzicky je složen z několika částí – pater.

Abstraktní třída Patro

Moderní typy úlů jsou složeny z více částí – komponent. Abstraktní entita `patro` představuje tyto komponenty. Těch může být několik typů, každý mající jiný účel. Koncept nástavkového úlu je vyvíjen po více než 150 let [13], tudíž nejsou předpokládány změny v jeho základní struktuře. Níže je uvedena konečná množina komponent, ze kterých se úl může skládat.

Obr. 3.2 Grafické ztvárnění komponent úlu: **1** – víko, **2** – krmítko, **3** – mateří mřížka, **4** – nástavek, **5** – úlové dno se zeleně vyznačeným podmetem.



Třída Dno

Základ úlu tvoří úlové dno. Po včely bývá zpravidla jediným přístupovým bodem do úlu. U instancí této třídy je důležité evidovat stav podmetu, jímž lze ovlivňovat proudění vzduchu v úlu.

Třída Nástavek

Nástavek představuje pro včely obytnou část úlu. Jsou v něm umístěny rámky s plástvemi, na nichž včely budují své dílo. Objekty rámků nejsou v systému kvůli přílišné detailnosti modelovány, jejich funkci zastupuje třída `Nástavek`. Obsahem rámků, abstraktněji nástavku, může být:

- *plod* – tvoří jej vajíčka nakladená matkou či vyvinutější larvy,
- *pyl* – včelami nasbíraný pyl z květů rostlin,
- *med* – získaný nektar včely přemění na med a ukládají do pláství,
- *zásoby* – cukerné zásoby podávané včelstvu pro překonání nepříznivého období,
- kombinace výše uvedeného; v takovém případě se uvádí převažující nebo významnější složka.

Smyslem nástavkového včelaření je přidávání, odebírání a přesouvání nástavků v průběhu roku, tudíž je přínosné evidovat i čísla nástavků k jejich rozpoznání.

Třída Mateří mřížka

Mateří mřížka je vkládána zpravidla mezi nástavky. Velikost otvorů v mřížce umožňuje bezproblémový průchod včel, ale zabraňuje průchodu včelí matky. Tak lze pohyb matky omezit a zabránit kladení vajíček mezi medové plásty. Instance této třídy nemají žádné parametry.

Krmítko

Instance této třídy mají velkou podobnost s instancemi třídy `Nástavek`. Na rozdíl od nástavku je v krmítku místo rámků umístěna nádoba s roztokem cukru, který je včelstvu dodáván pro překonání nepříznivých podmínek. Není potřeba evidovat číslo krmítka, v úlu bývá vždy nejvýše jedno.

Víko

Víko je nejsvrchnější komponentou celého úlu a tvoří tak jeho uzávěr. Instance této třídy nemají žádné parametry.

3.1.3 Uživatelské rozhraní evidenčního systému

Způsob ovládání počítačových programů pomocí grafického rozhraní je zcela přirozený a uživateli často vyžadovaný. Se zvětšujícím se počtem funkcí však přibývá i ovládacích prvků a zároveň se zmenšuje přehlednost programu. Je tedy vhodné méně často používané funkce přesunout do pozadí a zviditelnit jen několik nejpoužívanějších.

U navrhovaného evidenčního systému bude zřejmě patřit mezi nejpoužívanější funkce přidávání, odebírání a přesouvání nástavků a aktualizace jejich obsahu. Proto by tyto funkce měly být realizovatelné jedním kliknutím myši či stiskem klávesy. Podobně by měly být realizovány změna obsahu nástavku a zaplnění rámků. Při častém kočování je vhodné zobrazení umístění stanoviště na mapě a jeho jednoduchý přesun.

3.2 Návrh simulačního modelu medové snůšky

Model simulace medové snůšky by měl umožnit předpověď činnosti včel létavek při sběru nektaru. Účel takové simulace je zjistit a zobrazit *kam* včely létají a *kolik* nektaru přinesou. Obecná predikce medové snůšky je kvůli širokým hodnotám parametrů natolik nepřesná, že pravděpodobnost správné předpovědi je velmi malá. Delší doba předpovědi navíc znamená větší odchylku od skutečnosti. I kdyby se podařilo zohlednit všechny důležité okolnosti, medová snůška by stále silně závisela na umění, postupech a zkušenostech včelaře. Teoreticky lze předpovědět místa, kam pravděpodobně budou včely létat a kolik nektaru či pylu při tom nasbírají.

3.2.1 Parametry simulace medové snůšky

Množství parametrů majících vliv na medovou snůšku je mnoho. Některé, například průměrnou rychlost a objem medových váčků včely, množství nektaru v květech rostlin, lze s jistou přesností určit. Jiné parametry, jako počet včelou navštívených květů, doba hledání medové snůšky a interakci včel, lze zjistit pomocí pozorování chování včel. Ostatní parametry, například povětrnostní podmínky, počasí a teplotu, lze dopředu jen odhadovat. Při kombinaci těchto parametrů pak může vzniknout velké množství chyb a nepřesností.

Bylo by tedy vhodné uživateli simulace nabídnout dodatečnou úpravu některých parametrů. Nezbytností je nastavení základních často se měnících parametrů simulace, například počasí, sluneční aktivity, okolní teploty, stavu plodin, atd. Hodnoty jiných parametrů mohou být do jisté míry invariantní a jejich změna by uživateli neměla být přístupná.

Simulační model zohledňuje tyto parametry:

- počasí a teplotu okolí,
- sluneční aktivitu,
- množství včel v úlech,
- fyzické vlastnosti včel – objem medových váčků,
- pozorované schopnosti včel – rychlost včely, doba sběru nektaru,
- plodiny rostoucí v blízkosti úlů a stav jejich květenství, přičemž různé plodiny mají pro včely různou hodnotu.

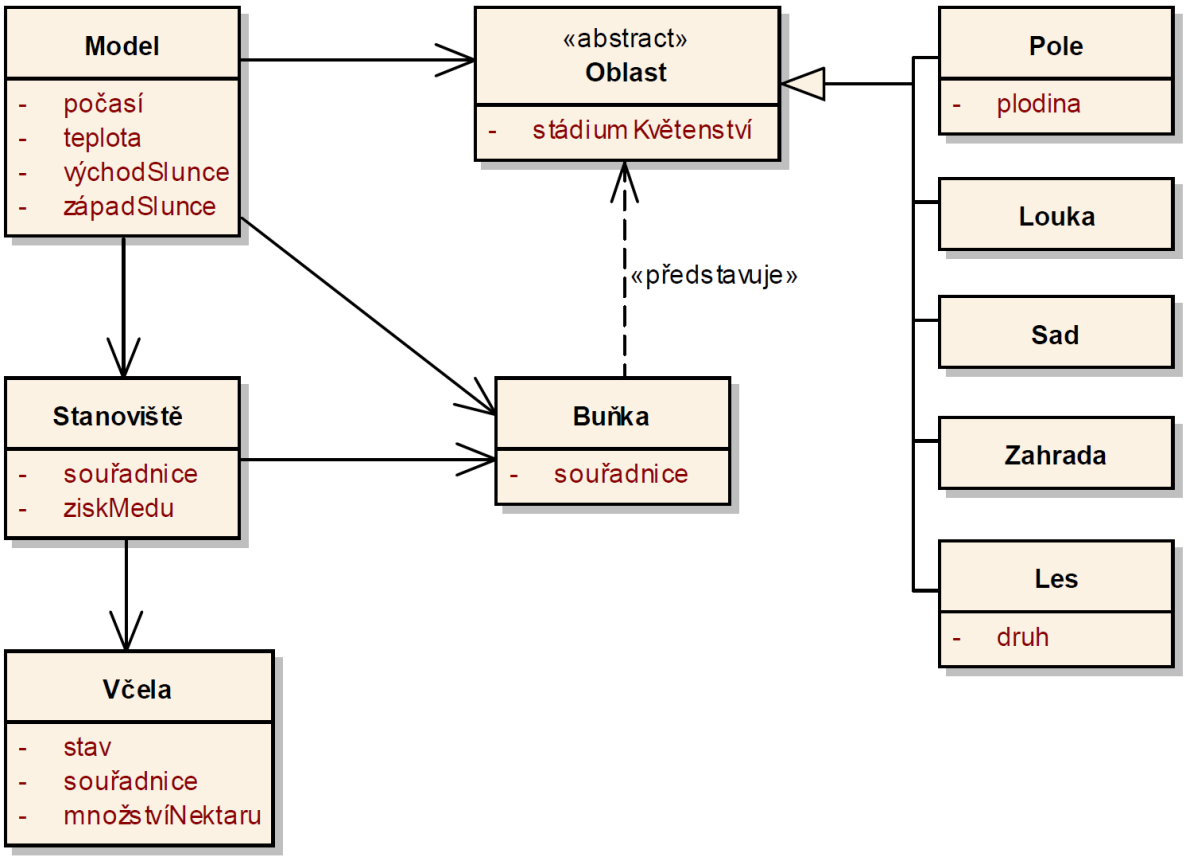
Ostatní parametry jsou zanedbány:

- skutečný stav včelstev,
- druh včel,
- úhel natočení úlů vůči slunci,
- konkrétní pozice úlů v terénu,
- vyhledávání snůšky včelami průzkumnicemi,
- předávání informací mezi včelami,
- kolísání teploty v průběhu dne,
- povětrnostní podmínky,
- a další.

3.2.2 Diagram tříd simulačního modelu

Abstrakcí reality lze získat několik objektů souvisejících se simulačním modelem – oblast, stanoviště, úl a včela. Jelikož pozice úlu v rámci stanoviště je zanedbatelná vůči poloze stanoviště v okolí, lze funkcionalitu množiny objektů představující úly přesunout do příslušného objektu stanoviště. Objekty představující úl se tak mohou zanedbat. K realizaci simulace je dále potřeba několik dalších typů objektů, které nemusí přímo souviset s objekty reálného světa, ale jsou přizpůsobeny architektuře výpočetního stroje.

Obr. 3.3 Diagram tříd simulačního modelu



Třída Model

V systému je vytvořena jediná instance této třídy. Umožňuje nastavení parametrů modelu před spuštěním jeho simulace a obsahuje další objekty podstatné pro simulaci.

Abstraktní třída Oblast

V okolí včelstev se mohou vyskytovat různé druhy rostlin a stromů tvořící oddělené oblasti. V těchto oblastech může růst specifický druh rostlin nebo více různých druhů. Jednotlivé druhy oblastí mají společné 2 skutečnosti: druh plodiny, popř. druhy plodin, a stádium květenství.

Třída Pole

Na polích se obvykle pěstují zemědělsky významné plodiny. Pro včelstva jsou tyto plodiny nesmírně významné a ve většině případů představují největší zdroj potravy. Každá z těchto plodin má pro včely jinou hodnotu. Tou může být kombinace množství obsaženého nektaru a jeho složení spolu s tvarem, hustotou a barvou květů.

Třída Louka

Louky představují místa, kde rostou převážné luční rostliny. V tomto modelu nejsou druhy luk dále děleny.

Třída sad

Za sad lze označit místo, kde jsou pěstovány ovocné stromy a keře. Obvykle sad obsahuje specifický druh stromů nebo keřů, v tomto modelu však pro zjednodušení nejsou dále specializovány.

Třída zahrada

Zahrady jsou podobné sadům s tím rozdílem, že je mimo výhradně ovocné stromy a keře mohou tvořit i další rostliny. Zahrady obvykle nebývají zaměřeny na specifický druh a jsou dosti různorodé, proto nejsou nijak dál děleny.

Třída les

Lesy lze rozdělit na jehličnaté, listnaté a smíšené. Z hlediska druhů medu se mezi významné stromy řadí lípa a trnovník akát. Zejména pak trnovník akát může vytvářet oblasti, kde rostou stromy výhradně tohoto druhu. Pro účely modelu jsou druhy lesů rozděleny na smíšené, lipové a akátové.

Třída Buňka

Kvůli potřebám výpočetního stroje je nutné rozdělit oblasti do menších částí. Mapa okolního prostředí úlů je tak rozdělena na malé čtvercové oblasti - buňky. Každá buňka kromě okrajových má 4 sousední buňky. Oblast může být tvořena více buňkami, přičemž každá buňka představuje právě jednu oblast.

Třída Stanoviště

Třída `Stanoviště` v simulačním modelu je podobná stejně pojmenované třídě v evidenční části systému. Instance této třídy v simulačním modelu však obsahují pouze základní informace o stanovišti – název stanoviště, jeho pozice a odhadovaný počet všech včel v úlech umístěných v tomto stanovišti.

Třída Včela

Instance této třídy představují včely. Pro účely simulace budou modelovány pouze včely létavky sbírající nektar. Včela se z hlediska medové snůšky může nacházet v několika stavech:

- `INIT` – stav inicializace, dle okolností simulátor určí, zda bude včela sbírat nektar či vůbec nevyletí z úlu,
- `NEAKTIVNI` – za nepříznivých podmínek včela zůstává v úlu a neúčastní se sběru nektaru**,**
- `ODLETA` – včela odlétá za snůškou rychlostí *6 až 8 m/s*,
- `SBIRA` – včela navštěvuje květy rostlin a přitom sbírá nektar, na každém květu stráví *5 až 15 sekund*,
- `PRILETA` – včela naplnila své medové váčky a vrací se s nektarem do úlu rychlostí *3 až 4 m/s*,
- `UKLADA` – včela se vrátila se snůškou do úlu, předává nektar ostatním včelám, a ten je ukládán do pláství po dobu *15 až 20 min*,
- `ODPOCIVA` – včela potřebuje chvíli na odpočinek, než se vydá za dalším sběrem nektaru, doba strávená odpočinkem činí *5 až 10 min*.

Výběr konkrétních hodnost je realizován pseudonáhodně s rovnoměrným rozložením pravděpodobnosti [21].

3.2.3 Návrh simulačního algoritmu

Algoritmus simulace je tvořen diskrétním simulátorem, který od času východu do času západu slunce v iteracích prochází stavy všech včel a určuje následující stavy dle kontextu. Nejprve jsou z evidenční části systému načteny informace o stanovištích a na základě velikosti včelstev je odhadnut počet létavek. Následně je spuštěna simulace.

Princip simulačního algoritmu je podobný reálnému modelu medové snůšky – včely vylétávají z úlu za snůškou, s nektarem se vrací zpět a předávají jej ostatním včelám. Odchylku představuje výběr cílové oblasti, jenž se neshoduje s realitou. Ve skutečném světě včely průzkumnice vyhledávají oblasti s cennými plodinami s určitou pravděpodobností nálezu nejvhodnější oblasti a s jistým časovým zpožděním. V simulačním systému je namísto toho vytvořen seznam nejvýhodnějších buněk v doletu včel a včely létavky jsou pak směřovány přímo na tyto buňky. Výběr cílových buněk je prováděn pseudonáhodně s Gaussovým rozložením pravděpodobnosti [21].

3.2.4 Vizualizace výsledků simulace

Simulace by měla v reálném čase zobrazovat informace o tom, do jakých oblastí včely létají a jaké množství nektaru jsou přitom schopny nasbírat. Množství nektaru je vhodné přepočítat na med, který z něj vznikne. Při implementaci dostatečně malých buněk lze vizualizaci provádět pomocí těchto buněk, například změnou barevného odstínu. Buňky, na nichž se nachází více včel, tak budou tmavší. Pokud by se graficky znázorňoval let včely, model by byl více výpočetně náročný a vizuálně méně přehledný.

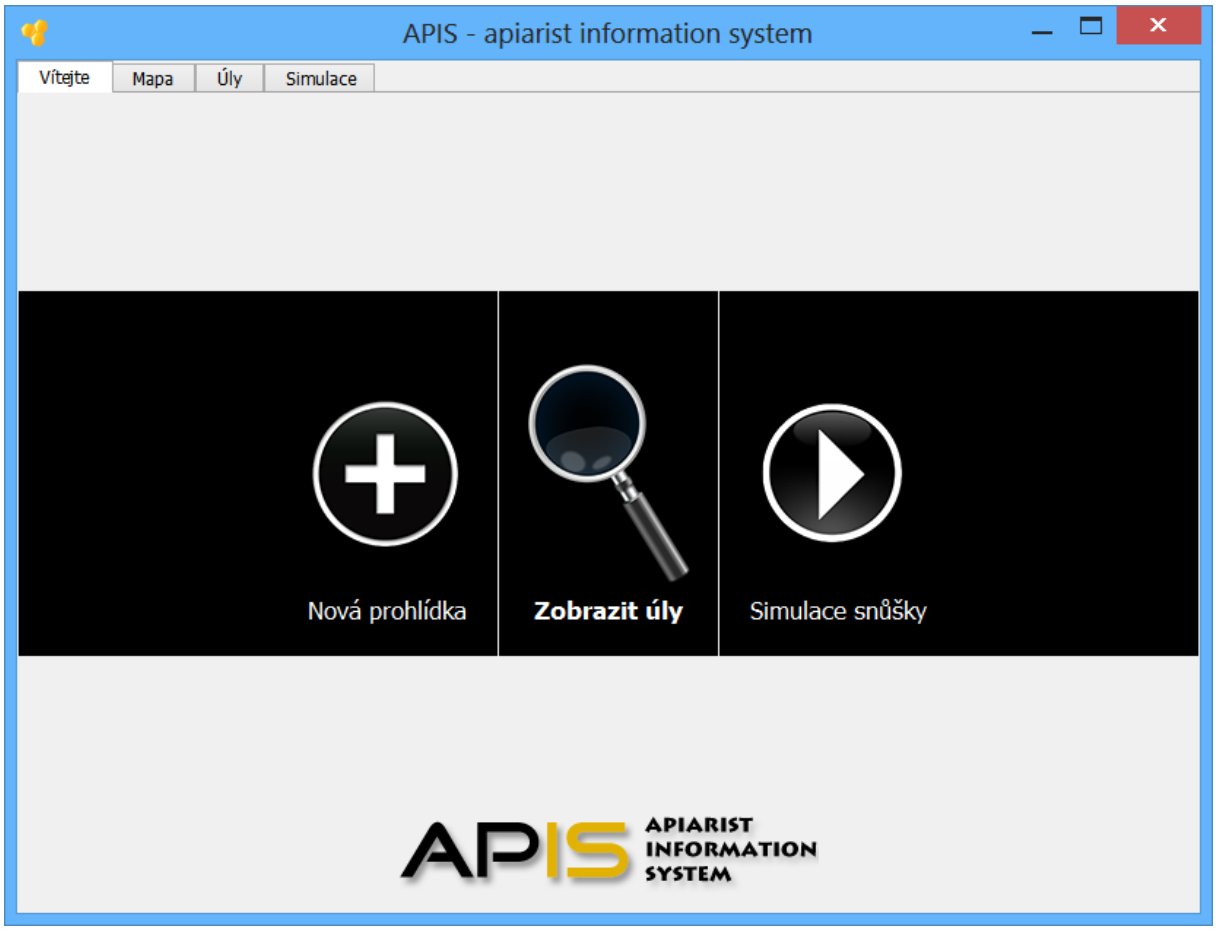
4 Implementace programu

Obě části, evidence stavu včelstev i simulace medové snůšky, byly vytvářeny v programech Qt Creator a Qt Designer [12]. Implementačním jazykem je C++ s využitím knihovny Qt. Minimální požadavky pro překlad programu jsou Qt verze 5.0.1 a překladač MinGW 4.7. Vývojové prostředí Qt podporuje multiplatformní vývoj aplikací, lze tedy i tento projekt přeložit a spustit na více platformách. Funkčnost překladu a běhu programu byly testovány na systémech Windows XP SP3, Windows 7 SP 1, Windows 8 a Linux Ubuntu 12.04. Aplikace je vyvíjena pod licencí LGPL a tato licence musí být zachována.

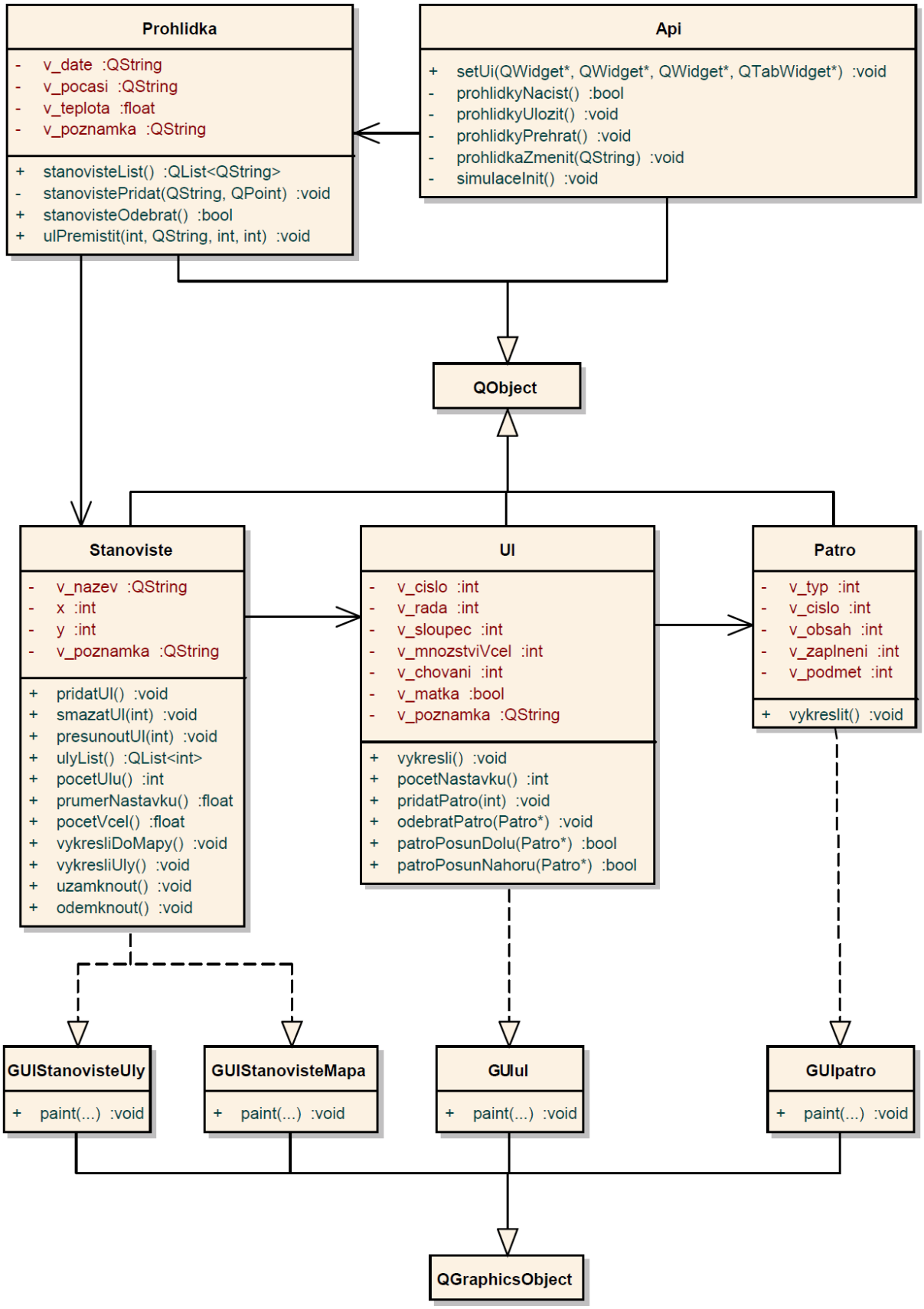
4.1 Evidence stavu včelstev

Vzhledem k použitému vývojovému prostředí byly vytvořeny pomocné třídy s přeponou `GUI` realizující grafické zobrazení příslušných tříd: `Stanoviste`, `Ul` a `Patro`. Podtřídy třídy `Patro` implementovány nebyly, jejich funkcionalita byla převedena do třídy `Patro`. Dále byla vytvořena třída `Api`, která implementuje logiku a řízení toku celého programu a poskytuje ostatním objektům rozhraní k ovládacím prvkům okna programu. Struktura evidenčního systému je znázorněna na obrázku 4.2.

Obr. 4.1 Úvodní stránka aplikace



Obr. 4.2 Diagram tříd implementovaných v evidenčním systému



4.1.1 Třída Patro

Tato třída zastupuje tyto komponenty úlu:

- úlové dno,
- nástavek,
- mateří mřížku,
- krmítko,
- víko.

Každá z těchto komponent vyžaduje specifické parametry a funkce. Všechny musí třída `Patro` implementovat.

Proměnné

Mezi důležité proměnné patří:

- `typ` – udává typ komponenty: *úlové dno, nástavek, mateří mřížka, krmítko* nebo *víko*,
- `cislo` – číslo nástavku,
- `obsah` – obsah rámků v nástavku: *pyl, med plod, zásoby, souše, mezistěny* nebo obsah krmítka: *cukerný roztok, api invert či medocukrové těsto*,
- `zaplneni` – odhadované množství obsahu v nástavku nebo krmítku,
- `podmet` – stav podmetu v úlovém dnu: *uzavřený, otevřený, není vložen*.

Všechny proměnné mají modifikátor přístupu nastaven na `private`. Typ těchto proměnných je `int`, každá může nabývat příslušných hodnot definovaných v souboru `define.h`. Proměnná `typ` musí být vždy definována, což nemusí platit pro ostatní proměnné. Podmíněnost jejich definice závisí na hodnotě proměnné `typ`.

Konstruktory

Objekt `Patro` lze instanciovat 4 základními způsoby dle jeho reprezentace příslušné části úlu:

- `Patro(int typ, QObject *parent)` - mateří mřížka a víko,
- `Patro(int typ, int podmet, QObject *parent)` – dno,
- `Patro(int typ, int obsah, int zaplneni, QObject *parent)` – krmítko,
- `Patro(int typ, int číslo, int obsah, int zaplneni, QObject *parent)` – nástavek.

V každém konstruktoru je jako poslední parametr předáván ukazatel na typ `QObject` reprezentující rodiče vytvářeného objektu, který je povinný a využíván některými metodami.

Rozhraní

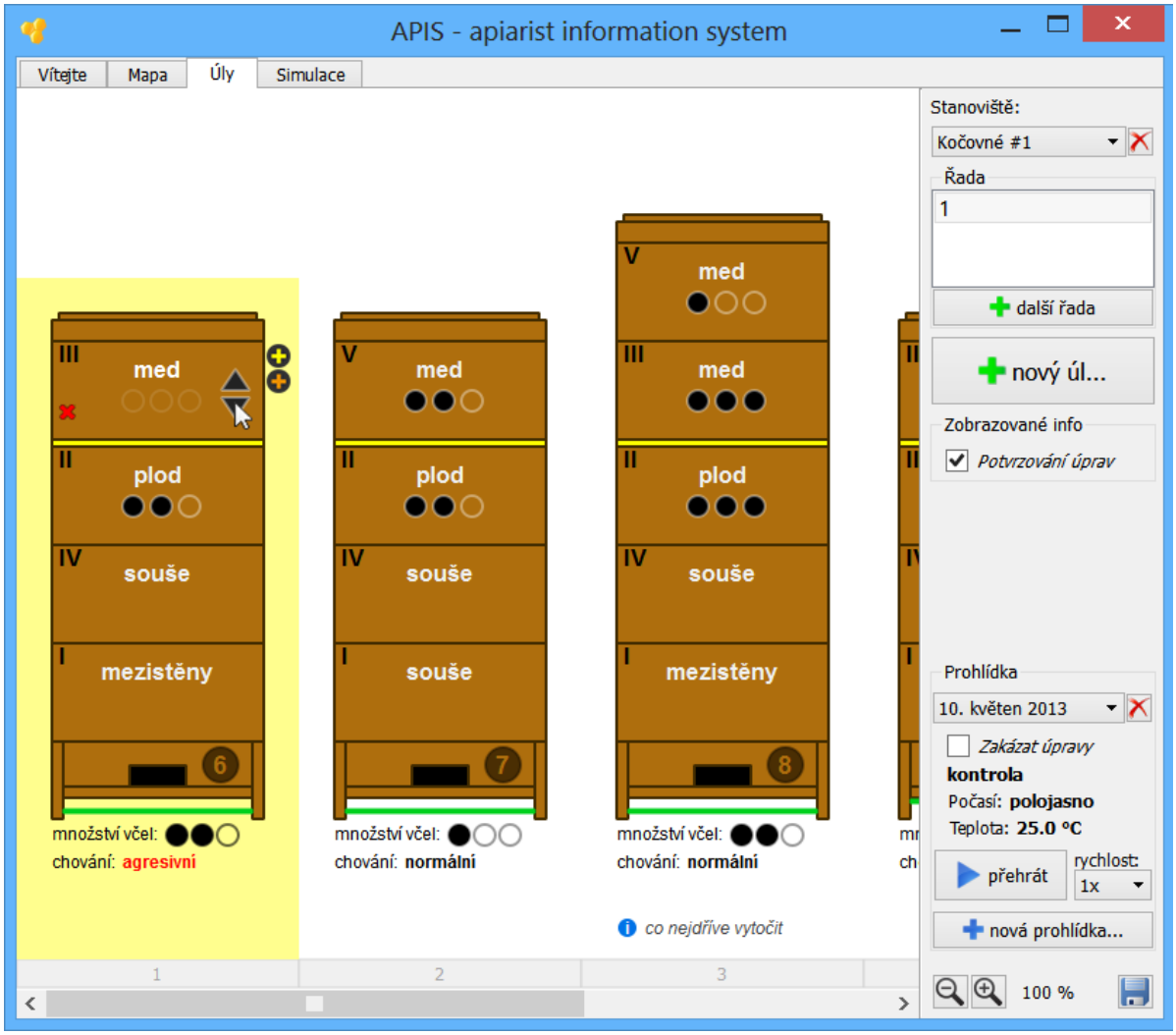
Mezi metody implementující rozhraní lze zařadit:

- `void zmenitCislo()` – zobrazuje formulář pro zadání nového čísla nástavku,
- `void obsah(int obsah)` – mění obsah nástavku či krmítka na jiný, v některých případech může nastavit na výchozí hodnotu proměnnou `zaplneni`,
- `void zaplneni(int zaplneni)` – nastavuje množství zvoleného obsahu v nástavku nebo krmítku,
- `void podmet(int podmet)` – mění nastavení podmetu v úlovém dnu,
- `void vykreslit()` – vykresluje aktuální stav patra, viz dále.

Vizualizace

Pro grafické ztvárnění objektu `Patro` slouží objekt třídy `GUIpatro`. Ten je vytvořen při volání metody `vykreslit()` a je mu předána reference na rodičovský objekt `Patro`. Skrz tuto referenci pak přistupuje k metodám rodičovského objektu a vykresluje aktuální stav *úlového dna, nástavku, mateří mřížky, krmítka* či *víka*.

Obr. 4.3 Ukázka funkcí nástavku



4.1.2 Třída Ul

Třída `Ul` zastupuje včelí úl jako celek. Umožňuje ukládat a aktualizovat informace o včelstvu.

Proměnné

Instance třídy `Ul` obsahují seznam komponent, ze kterých se úl skládá – objektů třídy `Patro`. Tyto objekty jsou v seznamu seřazeny dle svých skutečných pozic:

- `QVector<Patro*> patra`.

Mezi další důležité proměnné patří:

- `cislo` – číslo úlu, unikátní v rámci všech stanovišť,
- `chovani` – chování včel: **klidné, normální, agresivní**,
- `mnozstviVcel` – odhadované množství včel v úlu: **malé, střední, velké**,
- `matka` – logická hodnota udávající přítomnost matky ve včelstvu,
- `poznamka` – jakákoliv textová poznámka k úlu nebo včelstvu.

Proměnné udávají pozici úlu v rámci stanoviště:

- `rada` – udává řadu, ve které je úl umístěn,
- `sloupec` – pozice úlu zleva v rámci zvolené řady.

Všechny proměnné mají modifikátor přístupu nastaven na `private`. Proměnná `matka` je typu `bool`, `poznamka` typu `QString`, ostatní proměnné jsou typu `int`. Kombinace hodnot proměnných `rada` a `sloupec` jsou v rámci stanoviště unikátní.

Konstruktor

Jediným konstruktorem třídy `Ul` je:

- `Ul(int cislo, int rada, int sloupec, QObject *parent)`.

Objekt třídu `Ul` lze instanciovat pouze s poskytnutím parametrů `cislo`, `rada` a `sloupec`. Poslední parametr typu reference na `QObject` představuje ukazatel na rodiče vytvářeného objektu. Ostatní proměnné jsou nastaveny na výchozí hodnoty.

Rozhraní

Třída `Ul` implementuje některé metody pracující se seznamem `patra`:

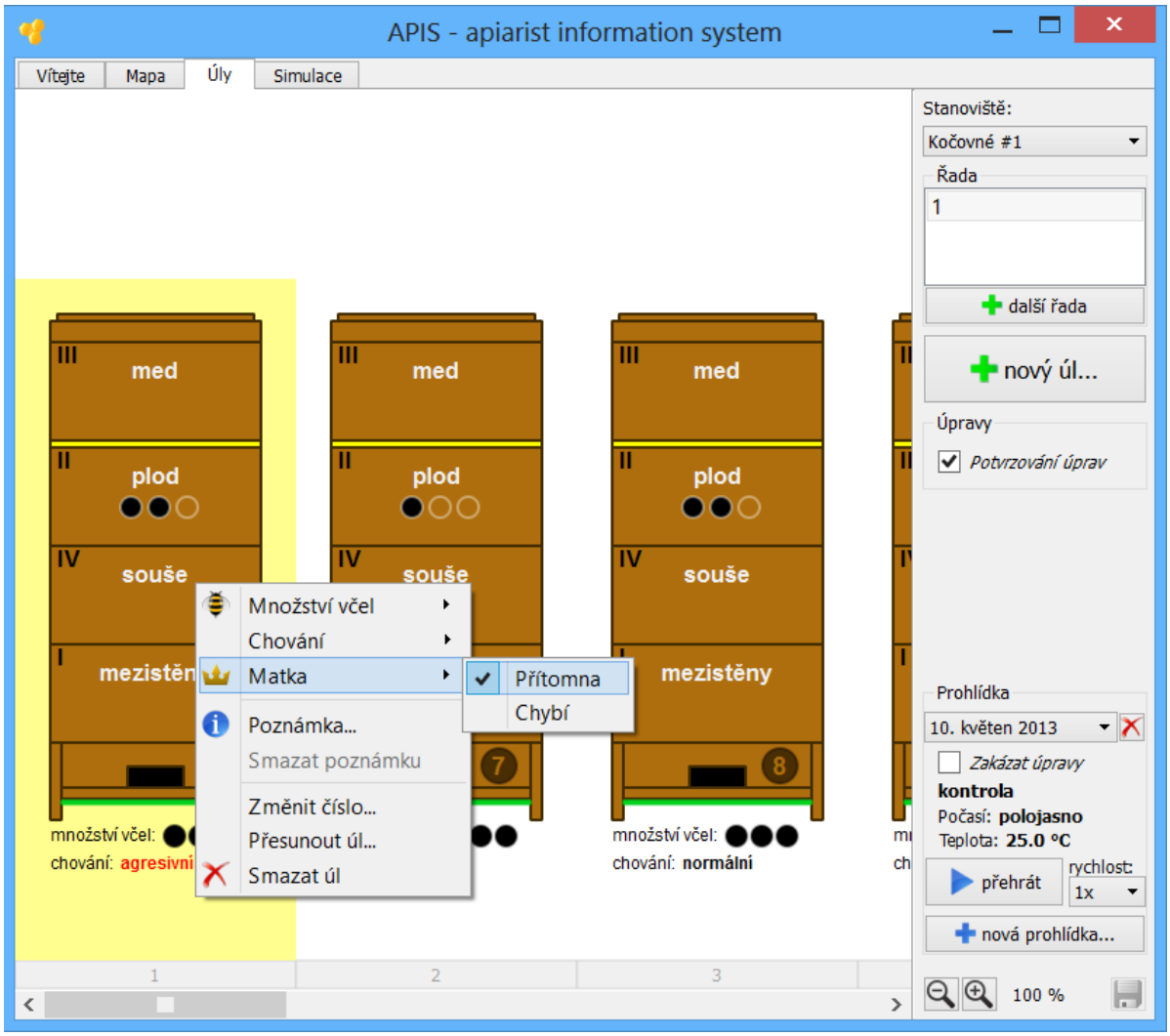
- `bool patroPosunNahoru(Patro *patro)` – vrací `true`, pokud bylo patro vyměněno s vrchnějším patrem, jinak vrací `false`,
- `bool patroPosunDolu(Patro *patro)` – vrací `true`, pokud bylo patro vyměněno se spodnějším patrem, jinak vrací `false`,
- `void patroPosunUplneNahoru(Patro *patro)` – posune zvolené patro co nejvíce nahoru,

- `void patroPosunUplneDolu(Patro *patro)` – posune zvolené patro co nejvíce dolů,
- `void pridatPatro(Patro *patro, Patro *podklad)` – vloží nové patro na zvolené, existující v úlu,
- `void odebratPatro(Patro *patro)` – odebere zvolené patro z úlu.

Další vybrané metody:

- `void zmenitMnozstviVcel(int mnozstviVcel)` – změní množství včel na uvedenou hodnotu,
- `void zmenitChovani(int chovani)` – změní chování včel na uvedenou hodnotu,
- `zmenitStavMatky(bool matka)` – nastaví přítomnost matky,
- `zmenitPoznamku()` – zobrazí formulář pro změnu textu poznámky,
- `smazatPoznamku()` – smaže text poznámky,
- `vykreslit()` – vykreslí aktuální stav úlu včetně všech jeho součástí.

Obr. 4.4 Nastavení parametrů úlu



Vizualizace

Grafické ztvárnění objektu `Ul` zajišťuje objekt třídy `GUIpatro`. Objekt třídy `Ul` je nosičem informace o včelstvu a úlu jako celku, tudíž je důležité tyto informace vhodným způsobem zobrazit uživateli.

4.1.3 Třída Stanoviste

Třída `Stanoviste` zastupuje skupinu úlů umístěných ve stejné lokaci.

Proměnné

Významnou proměnnou je tedy seznam všech úlů spadajících pod toto stanoviště:

- `QVector<Ul*> uly`.

Dalšími proměnnými jsou:

- `nazev` – unikátní název stanoviště,
- `x` a `y` – souřadnice umístění stanoviště,
- `poznamka` – textová poznámka je konkrétnímu stanovišti.

Všechny proměnné mají modifikátor přístupu nastaven na `private`. Proměnné `nazev` a `poznamka` jsou typu `QString`, souřadnice `x` a `y` typu `int`.

Konstruktor

Nový objekt třídy `Stanoviste` lze instanciovat pouze při splnění podmínky na unikátnost svého názvu v rámci všech stanovišť. Lze zadat i souřadnice stanoviště – to ale není nutné, souřadnice mohou být nastaveny později. Povinným parametrem reference na rodiče objektu typu `QObject`.

- `Stanoviste(QString nazev, QObject *parent, int x = 0, int y = 0, QString poznámka = QString())`

Rozhraní

Třída `Stanoviste` implementuje některé metody pro práci se seznamem `uly`:

- `void pridatUl()` – vložení nového úlu na zadanou pozici,
- `void smazatUl(int cislo)` – smazání úlu se zadaným číslem,
- `void presunoutUl(int cislo)` – přesunutí úlu na novou pozici či do jiného stanoviště.

Další vybrané metody:

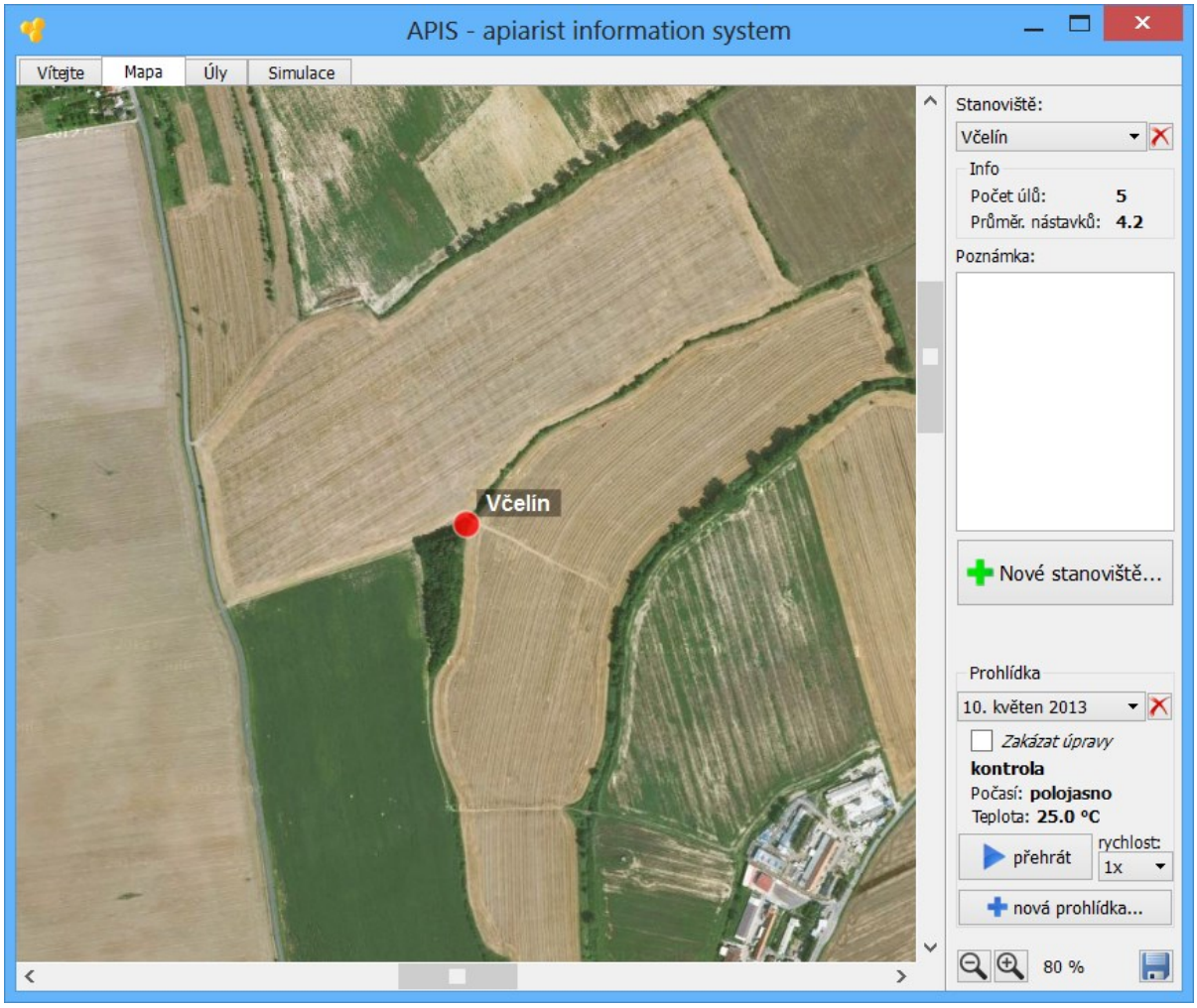
- `void vykreslitUly()` – vykresluje úly ve stanovišti umístěné v aktuální řadě,
- `void vykreslitDoMapy()` – vykresluje na mapě umístění stanoviště spolu s jeho názvem.

Vizualizace

Objekt `Stanoviste` lze vizualizovat dvěma způsoby:

1) *vizualizace stanoviště jako celku* – objekt `GUIStanovisteMapa` vykreslí stanoviště jako bod na mapě spolu s názvem dle jeho souřadnic,

Obr. 4.5 Vizualizace stanoviště na mapě



2) *vizualizace jednotlivých úlů* – objekt `GUIstanovisteUly` vykresluje pouze podstavce úlů zobrazující číslo jejich pozice a konkrétním úlům je zaslán požadavek na vykreslení.

4.1.4 Třída Prohlidka

Každá instance třídy `Prohlidka` uchovává specifický stav stanovišť a úlů k určitému datu.

Proměnné

Objekt třídy `Prohlidka` obsahuje seznam všech evidovaných stanovišť:

- `QVector<Stanovite*> stanoviste`.

Jedinou povinnou proměnnou je:

- `datum` – v rámci prohlídek unikátní hodnota – v jeden den může být vytvořena maximálně jedna prohlídka.

Ostatní proměnné jsou volitelné:

- `poznamka` – účel provádění prohlídky, např. budící prohlídka, kontrola, atd.,
- `pocasi` – počasí při provádění prohlídky,
- `teplota` – teplota při provádění prohlídky.

Všechny proměnné mají modifikátor přístupu nastaven na `private`. Proměnná `teplota` je typu `float`, ostatní proměnné jsou typu `QString`.

Konstruktor

Při vytváření prvního objektu třídy `Prohlidka` v systému je nutno zadat datum prohlídky a referenci na rodičovský objekt:

- `Prohlidka(QString datum, QObject *parent, QString poznamka = QString(), float teplota = 0.0, QString pocasi = QString()).`

Pokud v systému existuje alespoň jedna prohlídka, nová prohlídka se vytvoří hlubokým klonováním objektu poslední prohlídky. Pak může být nová prohlídka dále upravována.

Rozhraní

Třída `Prohlidka` implementuje především metody pro práci se stanovišti:

- `void stanovistePridat(QString nazev, QPoint bod)` – vytvoří nový objekt třídy `Stanoviste` se zadaným názvem a souřadnicemi a přidá jej do seznamu stanovišť,
- `void stanovisteOdebrat()` – odebere aktuální stanoviště.

Vizualizace

Kromě vizualizace aktuálního stanoviště samotná prohlídka zobrazuje jen textové informace o počasí, teplotě a účelu této prohlídky.

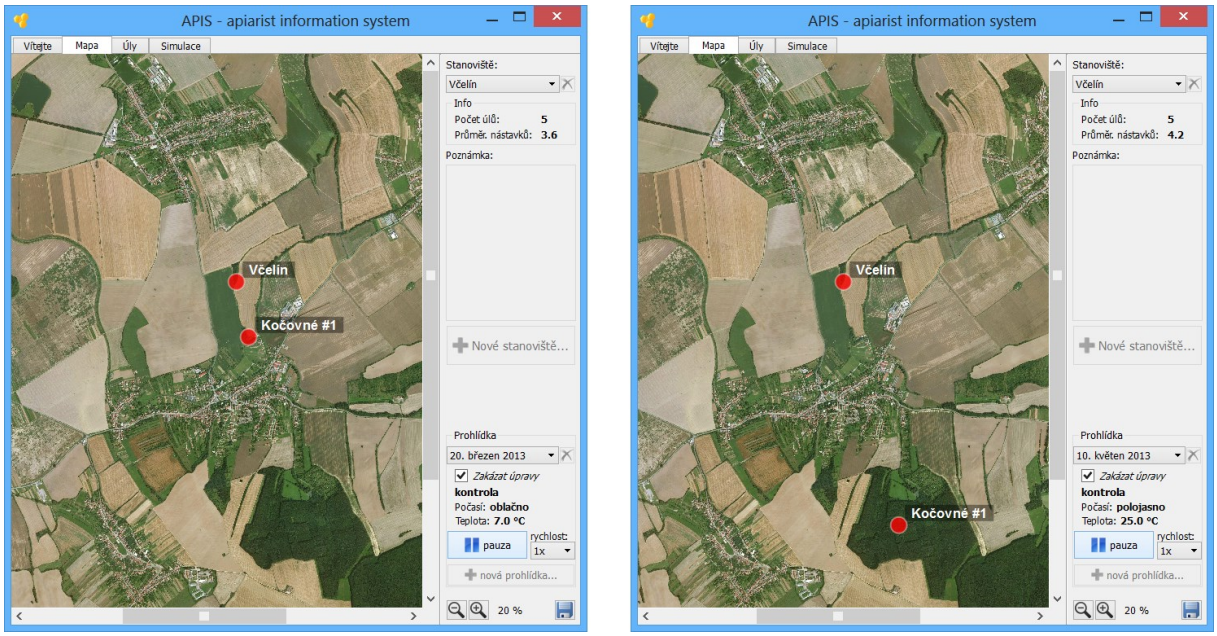
4.1.5 Třída Api

Třída `Api` je dosti odlišná od výše uvedených tříd. Na rozdíl od nich přímo nesouvisí s evidencí stavu včelstev, ale tvoří rozhraní k ovládacím prvkům programu. Při spuštění programu tato třída inicializuje ovládací a grafické prvky. Metody, jež tato třída implementuje, slouží především k načítání, ukládání, přepínání a animaci prohlídek:

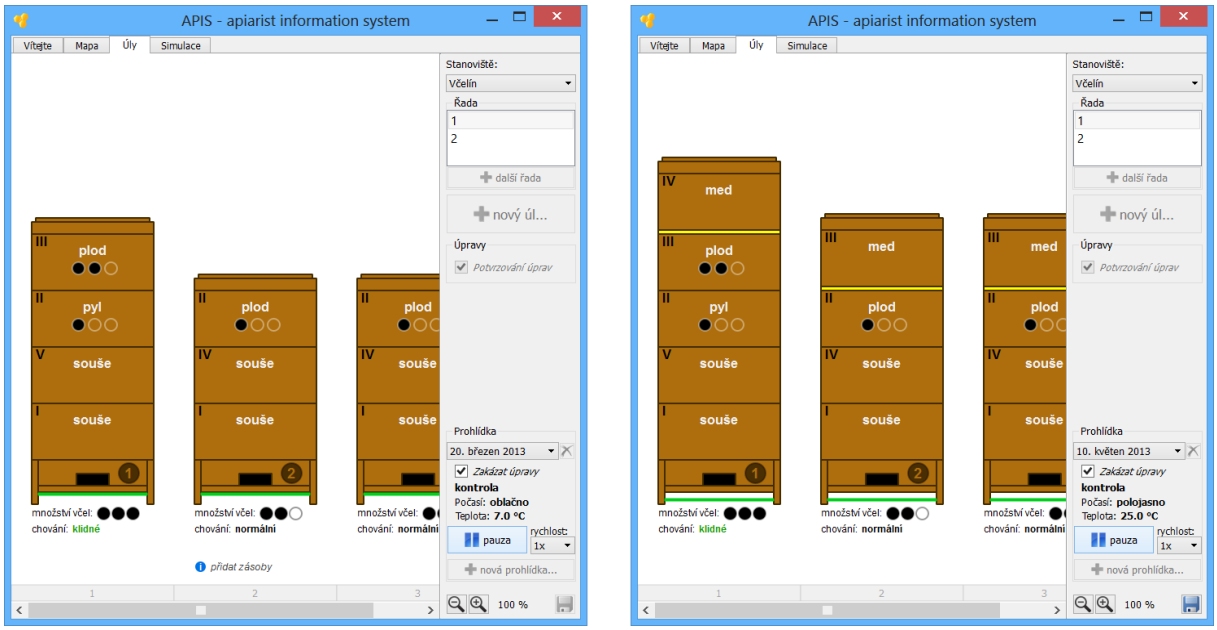
- `void prohlidkyUlozit()` – uložení všech prohlídek,

- `void prohlidkaOdebrat()` – odebrání konkrétní prohlídky,
- `void prohlidkaZmenit(QString datum)` – zobrazení prohlídky zadaného data,
- `void prohlidkyPrehrat()` – spustí animaci prohlídek,
- `void prohlidkyPauza()` – zastaví animaci prohlídek.

Obr. 4.6 Ukázka přehrávání změn pozic stanovišť



Obr. 4.7 Ukázka přehrávání změn stavu úlů



Načítání a ukládání dat evidence

Třída `Api` také poskytuje metody pro načítání a ukládání jednotlivých prohlídek. Všechny evidované informace jsou uloženy v XML formátu. Při načítání dat jsou postupně vytvářeny objekty tříd

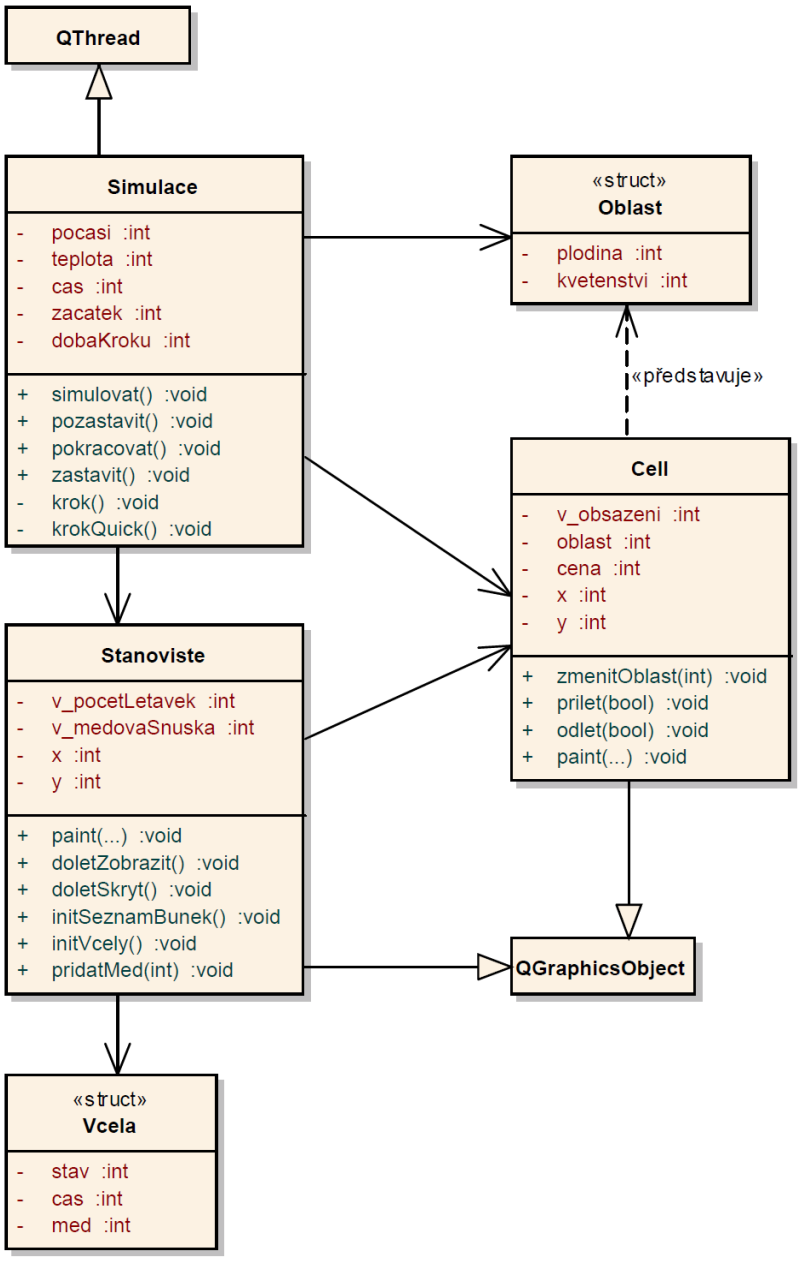
`Prohlidka`, `Stanoviste`, `Ul` a `Patro` dle jejich naposled uloženého stavu. Při ukládání jsou tyto objekty serializovány metodou `toElement` a uloženy ve formátu XML.

4.2 Simulace medové snůšky

Jelikož simulační část aplikace byla vyvíjena spolu s evidenčním systémem včelstev, její funkcionalita je do jisté míry s evidenčním systémem provázána. Obě části tvoří jednu aplikaci spojující funkce evidence a simulace včelstev. Z výše uvedených důvodů není simulační model implementován jako zásuvný modul, ale pevně zabudován do aplikace.

Simulace je optimalizována, aby byl její výpočet co nejrychlejší. Některé části tak nemusí zcela respektovat paradigma objektově orientovaného programování a blíží se spíše ke strukturovanému stylu programování.

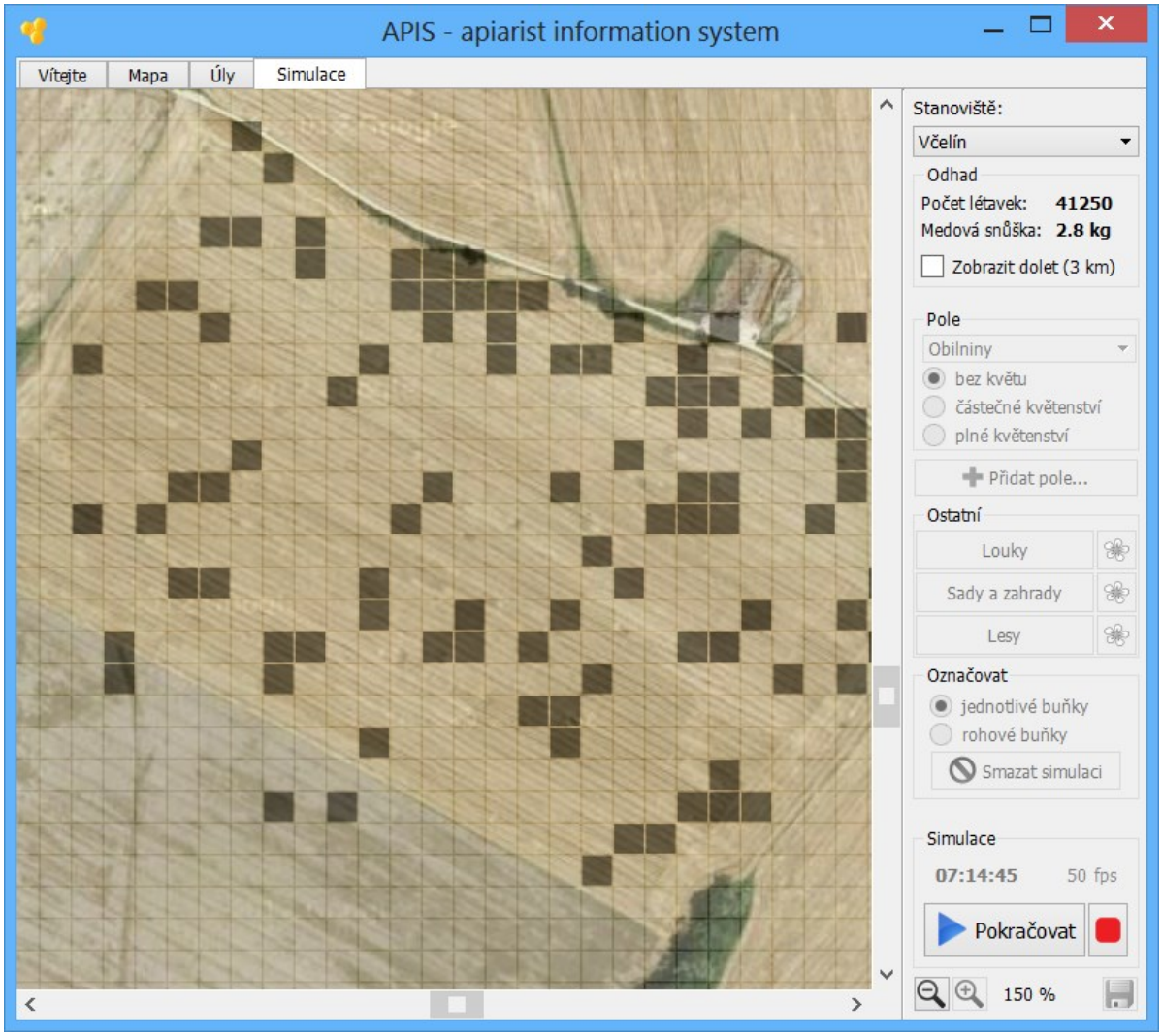
Obr. 4.8 Diagram tříd implementovaných v simulačním modelu



4.2.1 Třída Cell

Každá instance třídy `Cell` reprezentuje malou, v tomto modelu dále nedělitelnou část mapy. Mapa je rozdělena v pravoúhlé mřížce na několik desítek tisíc takových částí – **buněk**. Každá buňka má tvar čtverce o straně délky 14 pixelů odpovídajícím přibližně 23 m.

Obr. 4.9 Detail vizualizace buněk



Proměnné

- `oblast` – identifikační číslo oblasti, které buňka náleží,
- `x` a `y` – lokální souřadnice buňky.

Proměnné jsou typu `int` a kvůli optimalizaci je modifikátor přístupu nastaven na `public`.

Rozhraní

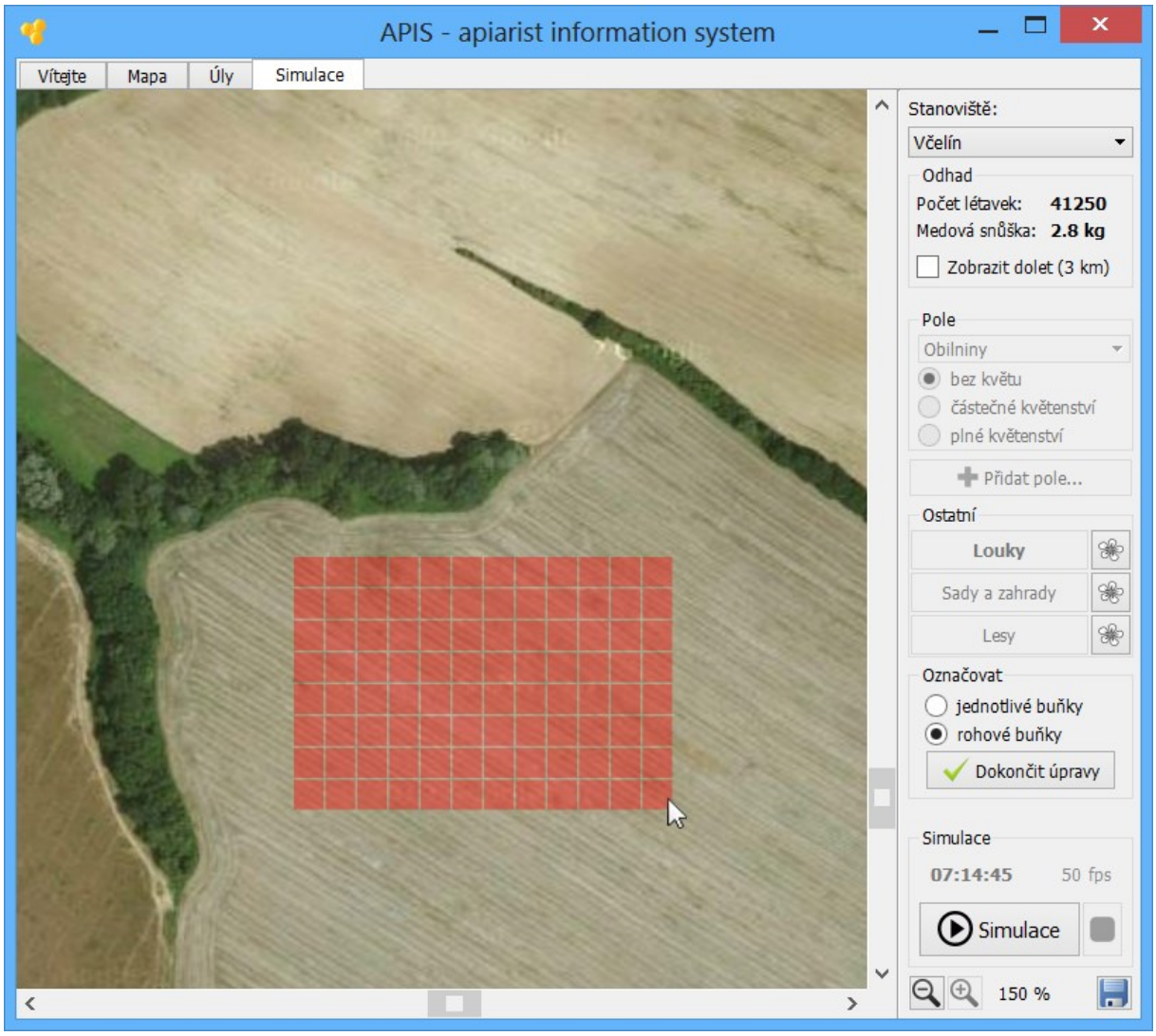
Mezi důležité metody patří zejména:

- `void prilet(bool vykreslit = true)` – značí přílet včely do této oblasti,
- `void odlet(bool vykreslit = true)` – značí odlet včely z této oblasti.

Účel těchto dvou metod je grafické znázornění aktuální lokace včely. Při příletu je obsah buňky ztmaven, při příletu zesvětlen. Další z implementovaných metod je:

- `void zmenitOblast(int oblast, bool aktivni = true)` – přiřadí buňku k nové oblasti. Výběr buněk probíhá buď **jednotlivě** – označení konkrétních buněk – **nebo hromadně** – označení rohových buněk.

Obr. 4.10 Ukázka výběru nové oblasti



4.2.2 Struktura Oblast

Informace o jednotlivých oblastech jsou uloženy ve statickém poli s identifikací tvořenou indexem do tohoto pole. Každá oblast obsahuje informace o *druhu* rostoucí plodiny a stadiu jejího *květenství*.

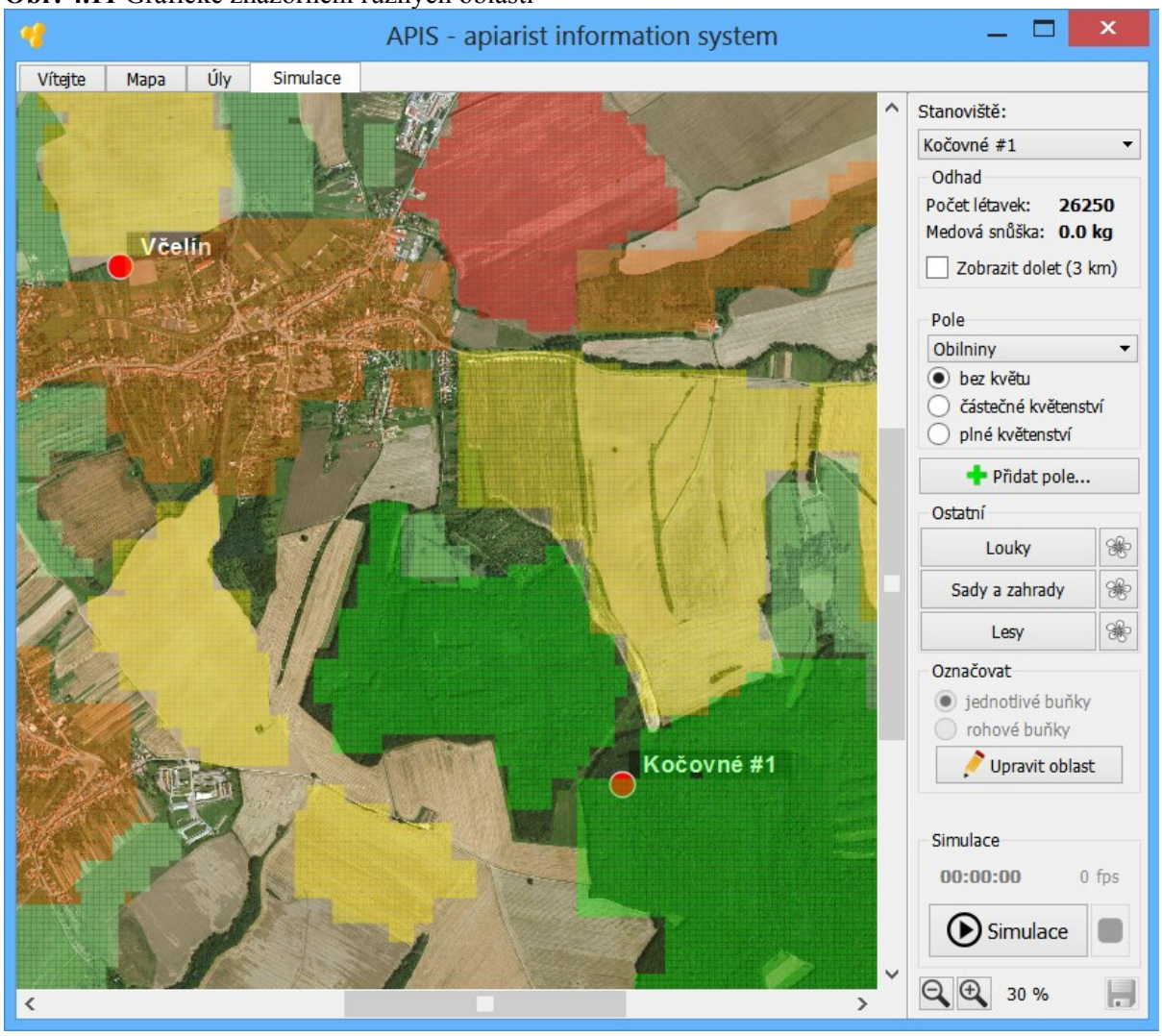
Plodiny

V modelu jsou evidovány různé typy oblastí, přičemž některé jsou blíže specifikovány:

- *pole* – specifikace: obilniny, řepka olejná, mák setý, slunečnice, pohanka nebo svazenka,

- *louky* – nejsou dále specifikovány,
- *sady a zahrady* – nejsou dále specifikovány,
- *lesy* – specifikace: smíšené nebo s dominantním druhem lípy či trnovníku akátu.

Obr. 4.11 Grafické znázornění různých oblastí



Květenství

Plodina se může nacházet v různých stadiích květenství, v modelu jsou zohledněny tyto:

- *bez květu* – plodina nekvete, tudíž neposkytuje žádný nektar,
- *částečné květenství* – plodina kvést zpravidla začíná nebo končí, množství poskytovaného nektaru ale nedosahuje nejvyšších hodnot,
- *plné květenství* – plodina poskytuje maximum svého nektaru.

Výpočet ceny buňky

Celková cena buňky sestává ze dvou částí: ceny plodiny a ceny vzdálenosti. Každé plodině je přiřazena vlastní cena s hodnotou 0 až 10:

- louky: 5,

- sady a zahrady: 6,
- lesy – smíšené: 2,
- lesy – lípa: 7,
- lesy – akát: 8,
- pole – obilniny: 2,
- pole – řepka olejná: 10,
- pole – mák: 4,
- pole – slunečnice: 8,
- pole – pohanka: 4,
- pole – svazenka: 9.

Dále je stanovena cena květenství plodiny s rozsahem hodnot 0 až 10:

- žádné květenství: 0,
- částečné květenství: 6,
- plné květenství: 10.

Celková cena plodiny se pak vypočítává vynásobením vlastní ceny plodiny a stadia květenství. Rozsahu hodnot této ceny se tak zvětšuje interval na 0 až 100. Cena vzdálenosti může nabývat hodnot 0 až 100, přičemž s rostoucí vzdáleností od úlu se tato hodnota lineárně zmenšuje. Jelikož vzdálenost snůšky má pro včely menší váhu než cena samotné plodiny, je celková cena buňky vypočítána jako: `cenaVzdálenosti + KOEFICIENT_PLODINY * cenaPlodiny`. Hodnota proměnné `KOEFICIENT_PLODINY` je nastavena na hodnotu 3.

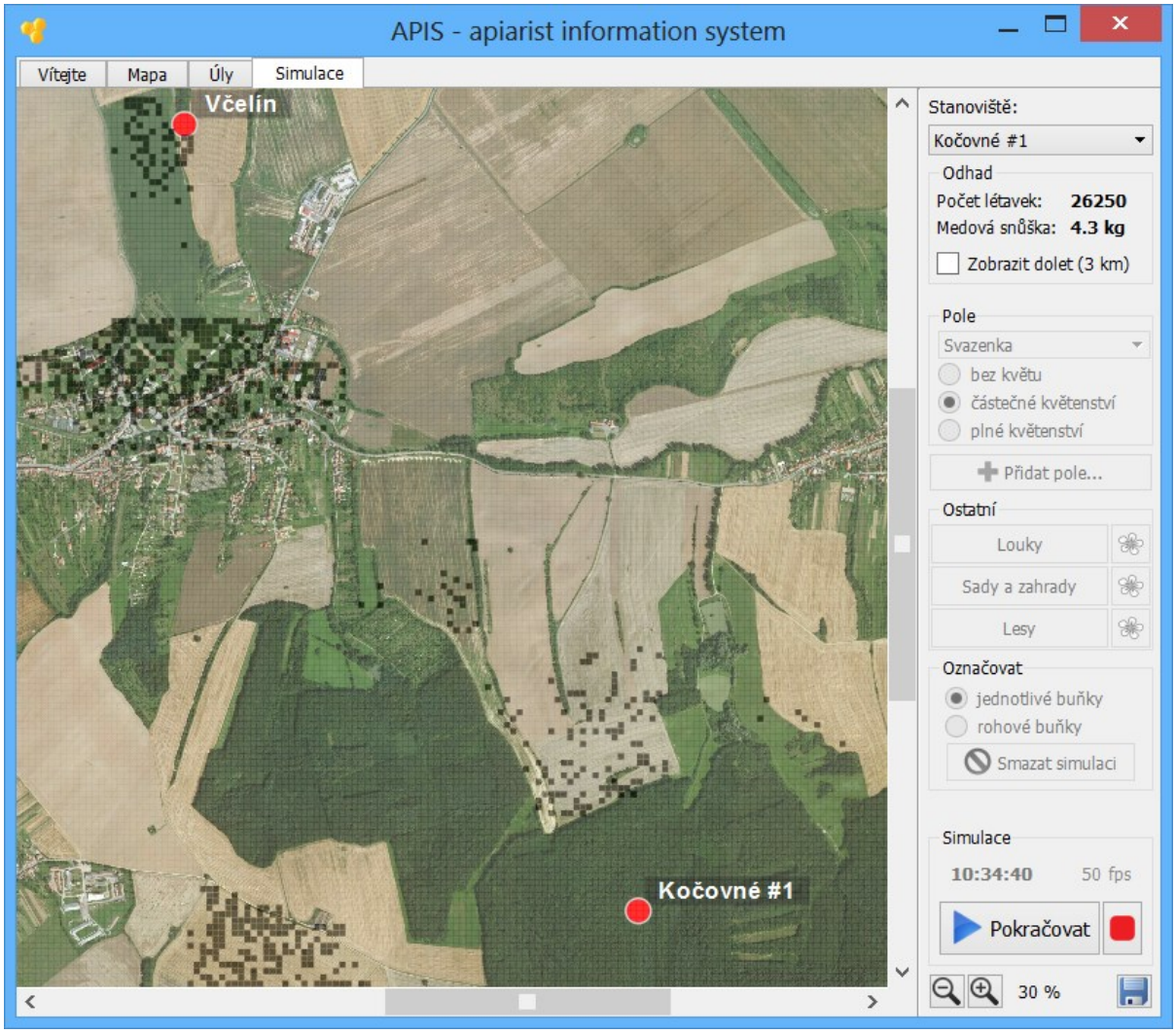
4.2.3 Struktura Včela

V simulačním modelu není z hlediska výkonnosti vhodné modelovat každou včelu létavku zvlášť. Struktura `Včela` tudíž představuje několik skutečných včel. Na tomto počtu také závisí schopnosti včely při sběru nektaru, které se přizpůsobí počtu zastupujících včel.

Proměnné

Kromě zmíněného stavu nese včela informace o **času**, po který bude vykonávat zadanou činnost – setrvávat v aktuálním stavu, a **nektaru**, který nasbírala.

Obr. 4.12 Vizualizace včel v simulaci



4.2.4 Třída Stanoviště

Na rozdíl od stejně pojmenované třídy v části evidence včelstev obsahují objekty této třídy jen některé základní informace.

Proměnné

- `x` a `y` – souřadnice stanoviště,
- `pocetLetavek` – odhadovaný počet létavek ze všech včelstev ve stanovišti,
- `medovaSnuska` – zaznamenává přírůstek medu během simulace,
- `QVector<Vcela*> vcely` – seznam simulačních včel létavek v daném stanovišti. Vzhledem ke skutečnému množství včel a nárokům na rychlost simulace tato simulační včela zastupuje několik desítek skutečných včel,
- `QVector<Cell*> okoli` – seznam okolních buněk stanoviště v doletu včel. Tyto buňky jsou před začátkem simulace seřazeny dle výhodnosti pro včely – vypočtené z ceny plodiny a vzdálenosti plodiny od stanoviště.

První 3 uvedené proměnné jsou typu `int` a jejich modifikátor přístupu je nastaven na `private`. Modifikátor přístupu dalších proměnných je nastaven na `public`.

4.2.5 Implementace simulačního algoritmu

Simulační algoritmus je implementován dle vzoru algoritmu diskrétního simulátoru. Od algoritmu obecného diskrétního simulátoru se implementovaný algoritmu liší v posouvání simulačního času. Z důvodu výkonnosti není vyhledáván nejbližší čas, kdy dojde ke změně stavu některé z komponent, ale stávající čas je inkrementován. Je zvolen tak malý časový krok, který na správnost konečných výsledků nemá vliv.

4.2.6 Spuštění simulace

Před začátkem simulace je nejprve vytvořen seznam okolních buněk pro jednotlivá stanoviště a pro každou buňku je vypočítána cena – viz kapitola Výpočet ceny plodiny. Podle této ceny je pak seznam seřazen. Dále je pro každé stanoviště vytvořen seznam simulačních včel.

Poté je spuštěn vlastní algoritmus simulace, který v iteracích prochází a určuje stavy všech včel ve stanovištích. Je-li začátek simulace nastaven na pozdější čas než je čas východu slunce, proběhne do požadovaného času tzv. rychlá simulace. V této části je v iteracích vypočítáván stav včel co nejvyšší rychlostí bez vykreslování scény. Po dosažení požadovaného času se simulace přepne do módu s vykreslováním scény a simulační rychlost je přizpůsobena požadovanému zrychlení. K blížícímu se času západu slunce přestávají včely vylétat z úlu a po jeho dosažení je simulace zastavena. Výsledky simulace jsou nadále zobrazeny. Jejich smazáním lze znovu nastavit oblasti či stanoviště a spustit simulaci s novými parametry.

5 Testování programu

Základní testy byly provedeny u obou částí programu – evidenční i simulační části. *Evidenční systém* byl testován především při svém vývoji, důraz byl kladen na jednoduchou a rychlou obsluhu. Na základě výsledků testů bylo grafické rozhraní několikrát upravováno až do své finální podoby. *Simulátor medové snůšky* byl testován s různým počtem včelstev. Tyto testy byly zaměřeny na rychlost simulace:

- při simulaci 20 silných včelstev rychlost simulace odpovídala požadovanému zrychlení,
- při simulaci 100 silných včelstev už bylo možno zpozorovat zpomalení simulace vůči nastavené rychlosti v řádu desítek až stovek sekund.

Proto byla zkoumána příčina tohoto zpomalení. Při vypnutí vizualizace opět simulace dosahovala požadované rychlosti. Z toho lze vyvodit, že zpomalení výpočtu bylo z velké části zapříčiněno vykreslováním scény, které není akcelerováno grafickou kartou.

5.1 Validita simulačního modelu

Verifikace simulačního modelu a následná validace je velmi obtížná z několika důvodů:

- většina hodnot použitých parametrů je širokých intervalů,
- model není založen ani inspirován žádnými existujícími validními simulačními modely,
- chování včel je samo o sobě mnohdy nepředvídatelné,
- nejsou zohledněny všechny okolnosti.

Dále model nedodržuje přesné chování včelstev – konkrétně při hledání snůšky. Včely průzkumnice totiž nemusí vždy najít tu nejvýhodnější a nejvýnosnější plodinu pro včelstvo nebo s několikadenním zpožděním. V tomto modelu jsou však ihned vybrány nejvýhodnější lokality, kam včely dále za snůškou létají. To se může odchylovat od skutečnosti, nicméně je tato odchylka menší, než kdyby bylo náhodně vybráno několik oblastí a z těchto vybrána ta nejvhodnější.

Návrh simulačního modelu byl založen na vlastních zkušenostech autora a odhadu některých parametrů. Například lze jen velmi těžce určit, jak pro včely klesá výhodnost plodiny s rostoucí vzdáleností od úlu – konkrétně stanovit hranici, kdy má výnosnější plodina vzdálenější od úlu stejnou cenu jako méně výnosná plodina blíže úlu. Méně složité je pozorováním včel určit, jaká plodina je pro ně oblíbenější. Pozorováním by také bylo možné zjistit, jak dlouho se včela zdrží na květu určitého druhu rostliny a kolik takových květů při jednom výletu navštíví.

Bohužel v době tvorby simulačního modelu nebylo možné tyto aktivity pozorovat, tudíž nejsou v modelu zohledněny. Stejně tak nebyla v praxi ověřena správnost modelu.

5.2 Možná vylepšení simulačního modelu

Simulační model by mohl být doplněn o více parametrů či rozšířeny stávající parametry. Především by mohl být do modelu zabudován algoritmus realizující hledání medové snůšky samotnými včelami a následně rozšířena doba běhu simulace z jednoho na více dnů. Pozorováním včel lze jistě upřesnit algoritmus výběru nejvhodnější plodiny v závislosti na její vzdálenosti od úlu. Dále by stávající oblasti mohly být rozšířeny o další plodiny, upřesněna průměrná doba strávená včelou na jednom květu příslušné plodiny a počet navštívených květů, hustota množství včel sbírajících nektar na určité ploše a mnohé další. V neposlední řadě by v modelu mohl být přesněji specifikován vliv počasí, teploty a povětrnostních podmínek na snůšku.

6 Závěr

Byla vytvořena aplikace pro podporu chovu včel umožňující evidenci informací o včelstvech a okolním prostředí a vytvořen simulační model umožňující simulaci medové snůšky. Evidence včelstev je zaměřena na vkládání a prohlížení stavu včelstev, s účelem poskytnout uživateli přehled nad těmito včelstvy. Simulační model je zaměřen na předpověď medové snůšky. Nastavením různých parametrů modelu lze simulovat letovou aktivitu včel při sběru nektaru.

Před návrhem aplikace byla provedena analýza dostupné literatury zabývající se problematikou včelařství. Dle zjištěných informací byla navržena aplikace tvořená dvěma částmi: systémem umožňujícím evidenci stavu včelstev a simulačním modelem medové snůšky. Aplikace byla implementována v jazyce C++ s využitím knihovny Qt a smí být dále upravována a šířena pod licencí LGPL.

Na závěr vývoje byly provedeny základní testy aplikace a diskutována validita modelu medové snůšky. Jelikož se nepodařilo zjistit hodnoty všech parametrů nebo identifikovat samotné parametry mající vliv na medovou snůšku, je verifikace vytvořeného simulačního modelu složitá a validita jen stěží dokazatelná. Přizpůsobením modelu konkrétní lokalitě a zohledněním přesného stavu včelstev lze verifikaci usnadnit. Validitu modelu lze zřejmě dokázat jen porovnáním výsledků simulace se skutečným chováním včel.

Před reálným využitím simulačního modelu k předpovědi medové snůšky je předpokládán jeho další vývoj. Simulační model může být doplněn o algoritmus vyhledávání medové snůšky a mohou být zohledněny další či upřesněny stávající parametry mající vliv na medovou snůšku.

Literatura

[1] Včelařské pojmy a zajímavosti. *Včelaři Konicko* [online]. [cit. 2013-04-01]. Dostupné z: http://vcelarikonicko.webnode.cz/vcelarske-pojmy-a-zajimavosti/

[2] Včely v číslech. *Pracovná spoločnost nadstavkových včelárov Slovenska* [online]. [cit. 2013-04-01]. Dostupné z: http://www.n-vcelari.sk/sal/VCELY88.html

[3] Matematika - včely a včelařství. *Včelky.cz* [online]. [cit. 2013-04-01]. Dostupné z: http://www.vcelky.cz/matematika.htm

[4] Kolik přinese medu jedna včela? *Fascinovaný včelař* [online]. [cit. 2013-04-01]. Dostupné z: http://ovcsvpardubice.blog.cz/0905/kolik-prinese-medu-jedna-vcela

[5] Může vás zajímat. *Včelařská farma Urban* [online]. [cit. 2013-04-01]. Dostupné z: http://www.vcelarskafarmaurban.cz/Zajimavo.htm

[6] *Apis - Apiary Information System* [online]. 2012 [cit. 2013-04-01]. Dostupné z: http://www.apissoftware.co.uk/

[7] Léčení moru včelího. *Fascinovaný včelař* [online]. [cit. 2013-05-13]. Dostupné z: http://ovcsvpardubice.blog.cz/0810/leceni-moru-vceliho-plodu-bez-leku

[8] PŘIDAL, Antonín. *Mor včelího plodu – diagnostika* [online]. [cit. 2013-05-02]. Dostupné z: http://user.mendelu.cz/apridal/text/c035.pdf

[9] Komunikace včel. *Pracovná spoločnost nadstavkových včelárov Slovenska* [online]. [cit. 2013-04-01]. Dostupné z: http://www.n-vcelari.sk/sal/VCELY48.html

[10] *Moderní včelař*. České Budějovice: PSNV-CZ, 2007, č. 2. ISSN 1214-5793.

[11] Komunikace včel. *Encyklopedie včel* [online]. [cit. 2013-05-13]. Dostupné z: http://www.n-vcelari.sk/sal/VCELY48.html

[12] *Qt Project* [online]. 2013 [cit. 2013-04-01]. Dostupné z: http://qt-project.org/

[13] VESELÝ, Vladimír. *Včelařství*. Vyd. 2., upr. a dopl. Praha: Brázda, 2003, 270 s. ISBN 80-209-0320-8.

[14] Simulation. *Wikipedia, The Free Encyclopedia.* [online]. [cit. 2013-05-11]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Simulation&oldid=554256974

[15] DEGRANDI-HOFFMAN, G., S.A. ROTH, G.L. LOPER a E.H. ERICKSON. *Ecological modelling: A honeybee population dynamics simulation model*. s. 133-150. ISSN 0304-3800. DOI: 10.1016/0304-3800(89)90088-4. Dostupné z: http://www.sciencedirect.com/science/article/pii/0304380089900884

[16] MARTIN, Stephen. *Ecological modelling: A population model for the ectoparasitic mite Varroa jacobsoni in honey bee (Apis mellifera) colonies*. s. 267-281. ISSN 0304-3800. DOI: 10.1016/S0304-3800(98)00059-3. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0304380098000593

[17] FATHIAN, Mohammad, Babak AMIRI a Ali MAROOSI. *Application of honey-bee mating optimization algorithm on clustering, Applied Mathematics and Computation*. s. 1502-1513. ISSN 0096-3003. DOI: 10.1016/j.amc.2007.02.029. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0096300307001853

[18] K-means. KUČERA, Jiří. [online]. [cit. 2013-05-12]. Dostupné z: http://is.muni.cz/th/172767/fi_b/5739129/web/web/kmeans.html

[19] CAMAZINE, Scott, James SNEYD. *A model of collective nectar source selection by honey bees: Self-organization through simple rules. Journal of Theoretical Biology*. s. 547-571. ISSN 0022-5193. DOI: 10.1016/S0022-5193(05)80098-0. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0022519305800980

[20] BAYER, Tomáš. Úvod do OOP. In: [online]. [cit. 2013-05-12]. Dostupné z: http://web.natur.cuni.cz/~bayertom/Prog2/prog2_1.pdf

[21] Základní typy rozdělení pravděpodobnosti spojité náhodné veličiny. [online]. [cit. 2013-05-12]. Dostupné z: http://homen.vsb.cz/~oti73/cdpast1/KAP05/PRAV5.HTM

[22] Langstroth Bee Lang. CUSHMAN A., David. [online]. [cit. 2013-05-12]. Dostupné z: http://www.dave-cushman.net/bee/lang.html

[23] PŘIDAL, Antonín. Mor včelího plodu. [online]. [cit. 2013-05-13]. Dostupné z: http://user.mendelu.cz/apridal/skripta/mor_f.htm

Seznam příloh

Příloha 1. CD

Příloha 1

Obsah CD

- source – zdrojové soubory nutné pro překlad aplikace,
- data – ukázková data sloužící k demonstraci funkcí aplikace,
- technicka_zprava – technická zpráva ve formátu pdf a docx.