

<https://vamuana.github.io/diplomovka/>

Diplomová práca – návrh a vývoj vzdelávacej aplikácie

PRÍRODNÍ BÁDATELIA

Bc. Vanesa Šoošová

Vedúci práce: doc. RNDr. Ľubomír Salanci, PhD.

POPIΣ PROBLÉMU A NAVRHNUΤÉHO RIEŠENIA

Interaktívno-vzdelávacích softvérov pre žiakov predškolského veku a prvých ročníkov ZŠ učiacich deti základy logického myslenia spojeného s programovaním v slovenskom jazyku je veľmi málo. Po prieskume ponúkaných softvérov sme však nenašli taký, ktorý by niektoré základné koncepty programovania uchopiteľné žiakmi tejto vekovej kategórie - cykly, podmienené príkazy, algoritmizáciu - spracovávali oddelene.

“Prírodní bádatelia” teda bude kombinovať vizuálne programovanie, príbeh a hru s rozdelenými konceptami na základe postavičiek. Každá postavička vo svojej zóne bude žiakov učiť svoj koncept formou gradovaných úloh, s možnosťou prepnutia medzi programovaním ikonkami a textovými príkazmi, softvér tak bude využiteľný pre väčšie rozpätie veku žiakov. Súčasťou práce bude výskum zameraný na použiteľnosť aplikácie v školskom prostredí a na základe späťnej väzby bude modifikovaná.

CIELE DIPLOMOVEJ PRÁCE

Hlavný cieľ:

Vytvoriť interaktívne edukačné prostredie, ktoré hravou formou učí základy programovania (sekvencie, cykly, podmienky) a je pedagogicky aj technicky použiteľné v triede.

Podciele:

- návrh edukačnej logiky a UX vhodného pre deti,
- návrh herného sveta a postavičiek,
- implementácia prototypu v Unity,
- vytvorenie blokového programátora,
- testovanie v školskom prostredí,
- analýza použiteľnosti.

VÝSKUMNÉ OTÁZKY

V1: Ako ovplyvňuje softvérová implementácia sekvencií, cyklov a podmienok používateľské správanie detí pri riešení úloh v interaktívnom prostredí?

V2: Ktoré UI komponenty a mechanizmy spätej väzby (highlighting, krokovanie, vizualizácia podmienok a iterácií) najviac podporujú efektívne riešenie úloh a minimalizujú chybovosť používateľa?

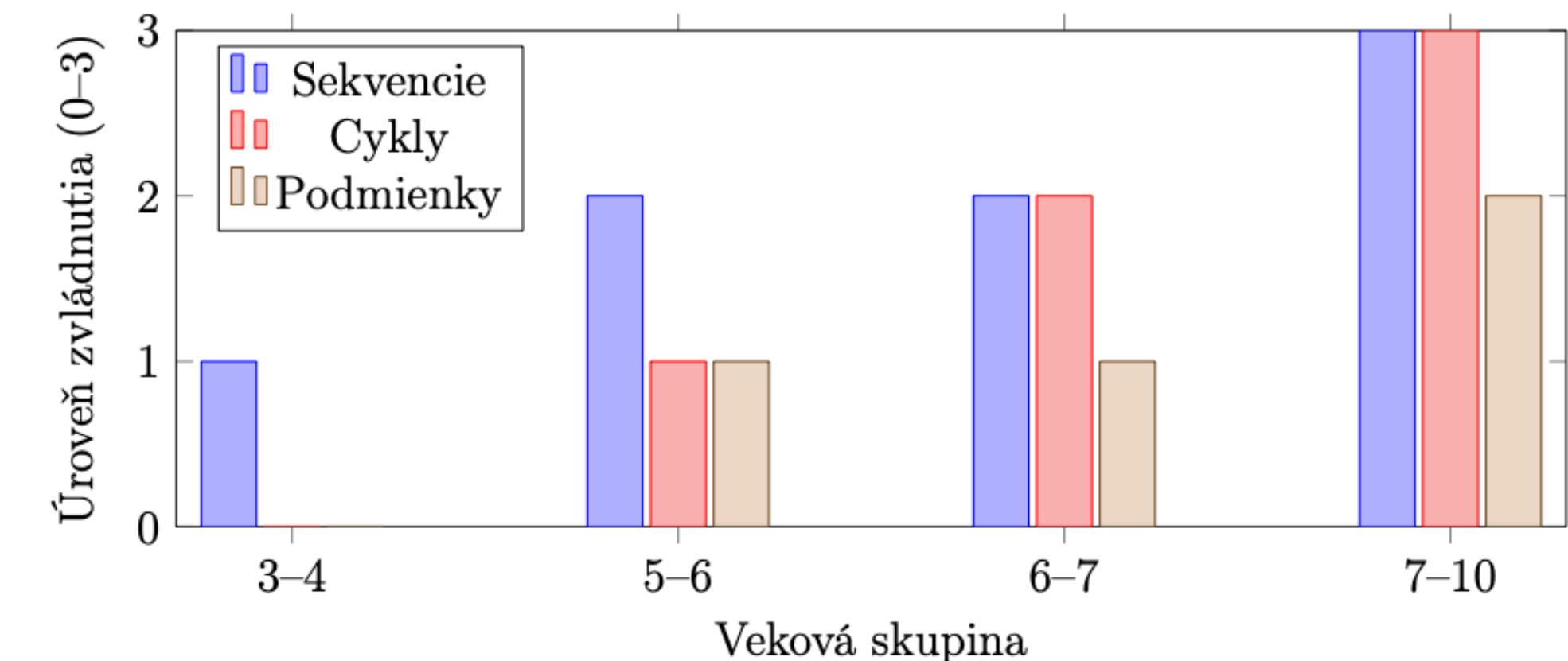
V3: Ako hodnotia pedagógovia technickú použiteľnosť, intuitívnosť a didaktickú vhodnosť softvéru z hľadiska jeho funkcionálít a návrhových rozhodnutí?

Hypotézy:

- Predpokladáme, že zo systémových logov je možné automaticky extrahovať opakujúce sa vzorce interakcie (interaction patterns), ktoré korelujú s úspešnosťou riešenia programu, a teda môžu slúžiť ako základ modelu študenta (student model).
- Predpokladáme, že UI mechanizmy poskytujúce okamžitú systémovú spätnú väzbu (highlight aktuálneho príkazu, vizualizácia iterácií cyklu, označenie miesta chyby) znižujú počet opakovanych interakcií s UI (dragov, úprav blokov, reštartov), čo sa prejaví nižšou frekvenciou UI eventov v logoch.
- Predpokladáme, že pri použití drag-and-drop mechanizmu bude celkový čas skladat' program kratší než pri alternatíve založenej na klikacích ponukách.

TEORETICKÉ VÝCHODISKÁ: COMPUTATIONAL THINKING

- Definícia CT
- Kľúčové komponenty CT:
 - sekvencie,
 - cykly,
 - podmienky,
 - dekompozícia,
 - debugovanie.
- Potvrdená postupnosť učenia:
sekvencie → cykly →
podmienky (kap. 2 práce)



Obr. 2.1: Syntetická schéma „vek vs. zvládnuté koncepty“ podľa prehľadov a empirických štúdií [3, 4, 11].

PREHĽAD EXISTUJÚCICH NÁSTROJOV

Zahraničné nástroje

1. ScratchJr +príbehy, jednoduché bloky, motivácia -obmedzené konštrukty, bez adaptácie, nie v slovenčine
2. BeeBot +hmatateľnosť, okamžitá spätná väzba – vysoké náklady, potreba fyzického priestoru
3. Lightbot +jasné CT princípy, hlavolamy -málo tvorivosti, menej vhodné pre 4-6 rokov
4. Kodable +systematické kurikulum - uzavretý ekosystém, platené
5. Blockly / Code.org +flexibilita, škálovateľnosť -príliš komplexné pre malé deti, málo príbehovost'

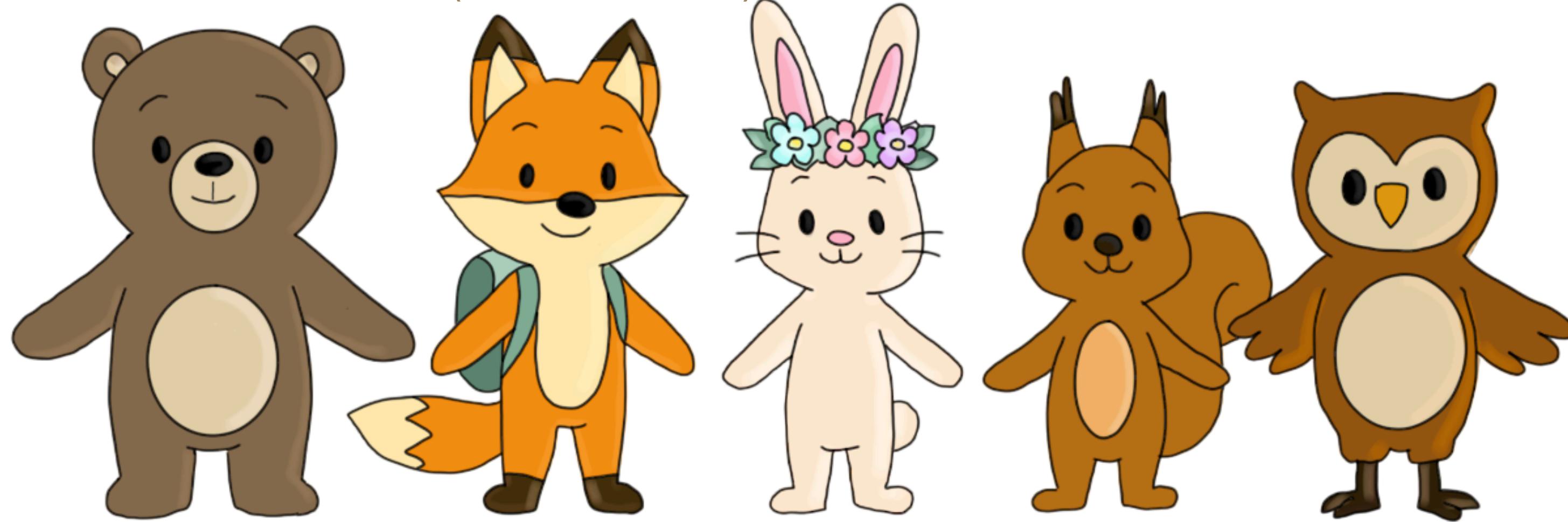
Slovenské nástroje

1. Emil na cestách / v cirkuse +jasná ikonografia, silná didaktika -neobsahuje blokové programovanie
2. Šašo Tomáš +multimediálne hravé úlohy -nie je zamerané na CT
3. IzyLogo +vizuálne programovanie (Logo) -nevzhodné pre 4–6 rokov, žiadny príbeh
4. Živý zošit +moderné multimodálne aktivity -nezamerané na CT ani programovanie

NÁVRH POSTÁV

Nasledoval výber príkazov, ktoré deti chceme naučiť, a následne tvorba návrhu hlavných postáv, ktoré predstavujú informatické koncepty:

- Líška: sekvencie
- Zajac: podmienky
- Veverička: cykly
- Sova: rekapitulácia a výzvy
- Medved': voľná tvorba (sandbox mód)

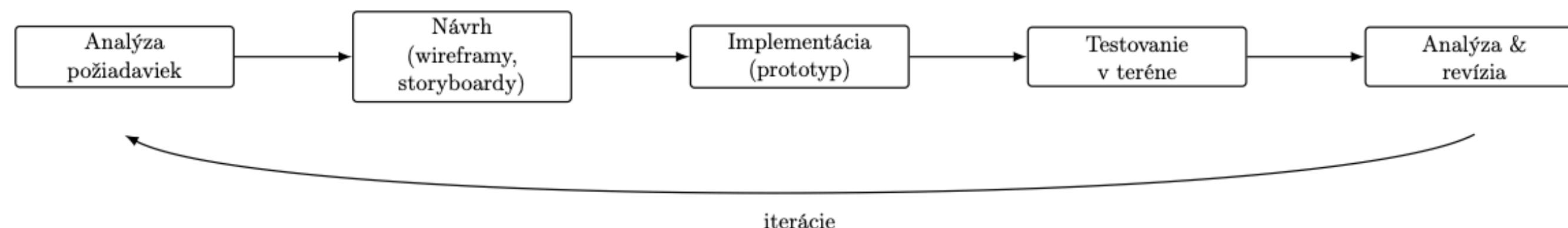


METODOLÓGIA: DESIGN-BASED RESEARCH (DBR)

Design-Based Research (DBR) je zvolený metodologický prístup, ktorý prepája návrh edukačného softvéru s jeho systematickým skúmaním v autentickom prostredí školy.

Kľúčové črty DBR (podľa kap. 3 práce):

- Iteratívny cyklus: analýza → návrh → implementácia → testovanie → revízia
- Prepojenie teórie a praxe - dizajn je ukotvený vo výskume CT, HCI a detskej kognície
- Testovanie v reálnych triedach - malé skupiny detí v MŠ/ZŠ1, pozorovanie, logy, rozhovory
- Participácia učiteľov - spätná väzba priamo z pedagogickej praxe
- Hypotetickosť prototypov - každá verzia je návrhom hypotézy, ktorá sa ďalej overuje



Obr. 3.1: Iteračný cyklus DBR pre Prírodných bádateľov.

PROCES VÝVOJA SOFTVÉRU

1. Analýza požiadaviek

- vek 4-10 rokov, kognitívne limity, predčitateľský stav
- typ zariadení (tablet/PC), spôsob práce v triede
- pedagogické ciele: sekvencie, cykly, podmienky
- výstup: persony, scenáre používania

2. Návrh prototypu

- wireframy UI, storyboardy scén
- definícia palety blokov a ich postupného odomykania
- návrh herných módov (tréning, dobrodružstvo, sandbox)
- návrh postavičiek a didaktických situácií

3. Implementácia prototypu

- Unity + C#, vlastný blokový editor
- modelovanie postáv a animácie v Blenderi
- interpreter príkazov, krokovanie, spätná väzba
- základ modelu študenta

4. Testovanie v teréne

- malé skupiny detí v MŠ/ZŠ1
- pozorovanie, UI logy, „mysli nahlas“, spätná väzba učiteľov
- identifikácia problémových miest

5. Revízia a úprava prototypu

- zmeny v UX, ikonách, blokoch, obtiažnosti
- nové úlohy, vylepšené animácie
- príprava ďalšej iterácie

UŽÍVATEĽSKÉ ROZHRANIE A OVLÁDANIE

Ciele návrhu

- Prehľadné dvojzónové rozhranie: program + scéna
- Minimálna kognitívna záťaž, konzistentné UI
- Okamžitá vizuálna spätná väzba (highlight, krokovanie, chyby)

Hlavné komponenty UI

- Program Workspace – drag & drop blokov, AST štruktúra
- Execution View – scéna, animácie, kolízie, vykonávanie
- Control Layer – play/stop/krokovanie/replay
- Feedback Layer – vizualizácia iterácií, chybové stavy



Interakčné vzory

- presúvanie blokov, úprava parametrov
- krokové vykonávanie programu
- zobrazenie aktívneho príkazu / vetvy / iterácie

ARCHITEKTÚRA UI A FUNKČNÉ SPRÁVANIE

Architektonické časti

- UI Layer - vizuálne komponenty
- Command Builder - prevod blokov → model
- Program Model (AST) - strom príkazov
- Interpreter Core - vykonávanie programu
- Scene Model - 2D mriežka, objekty, stavov

Tok vykonávania programu (funkčný diagram)

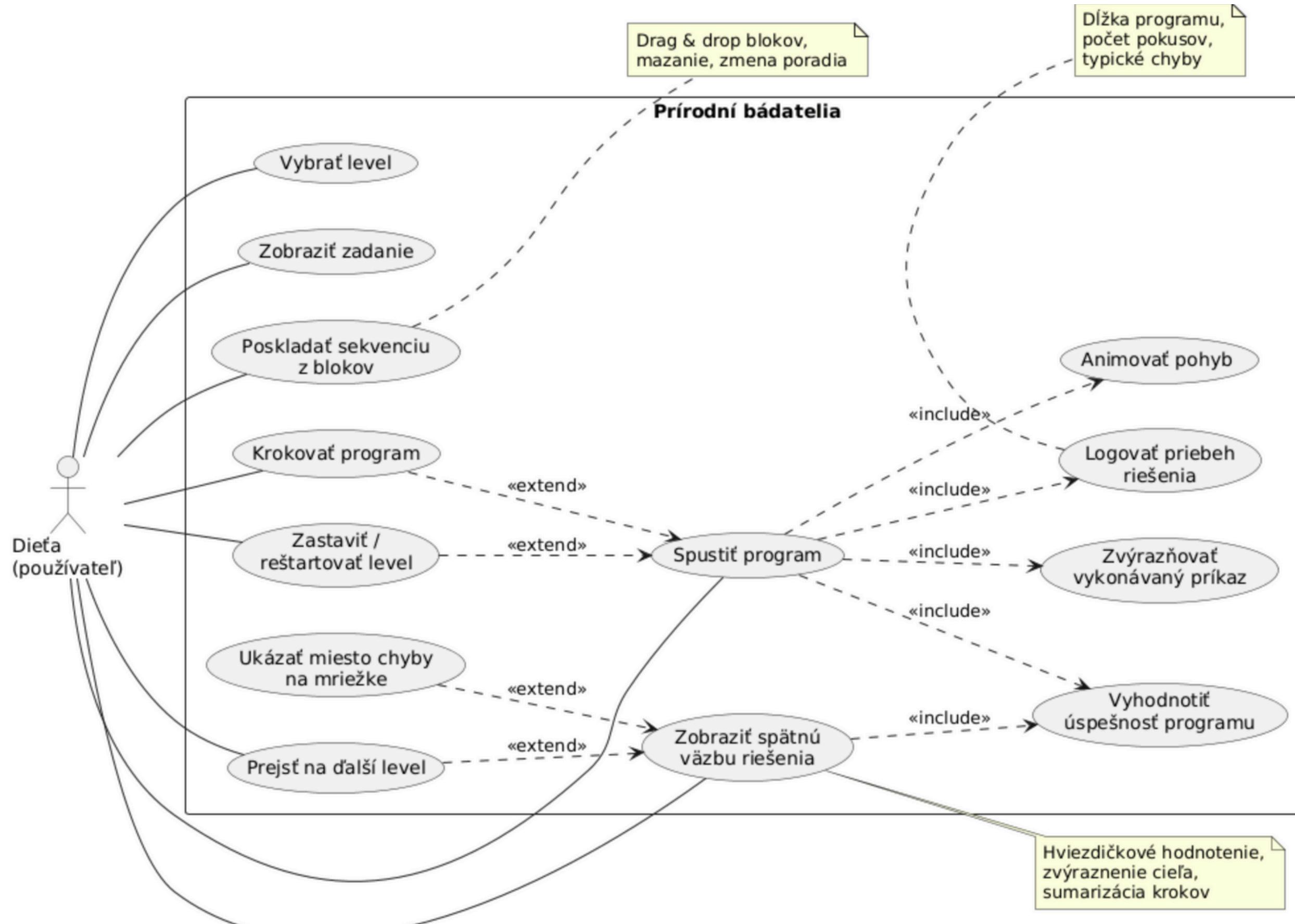
1. Používateľ zostaví bloky
2. Parser vytvorí AST
3. Interpreter prechádza uzly
4. Agent vykoná akcie podľa Scene Modelu
5. Vizualizácia + logovanie chýb a stavov



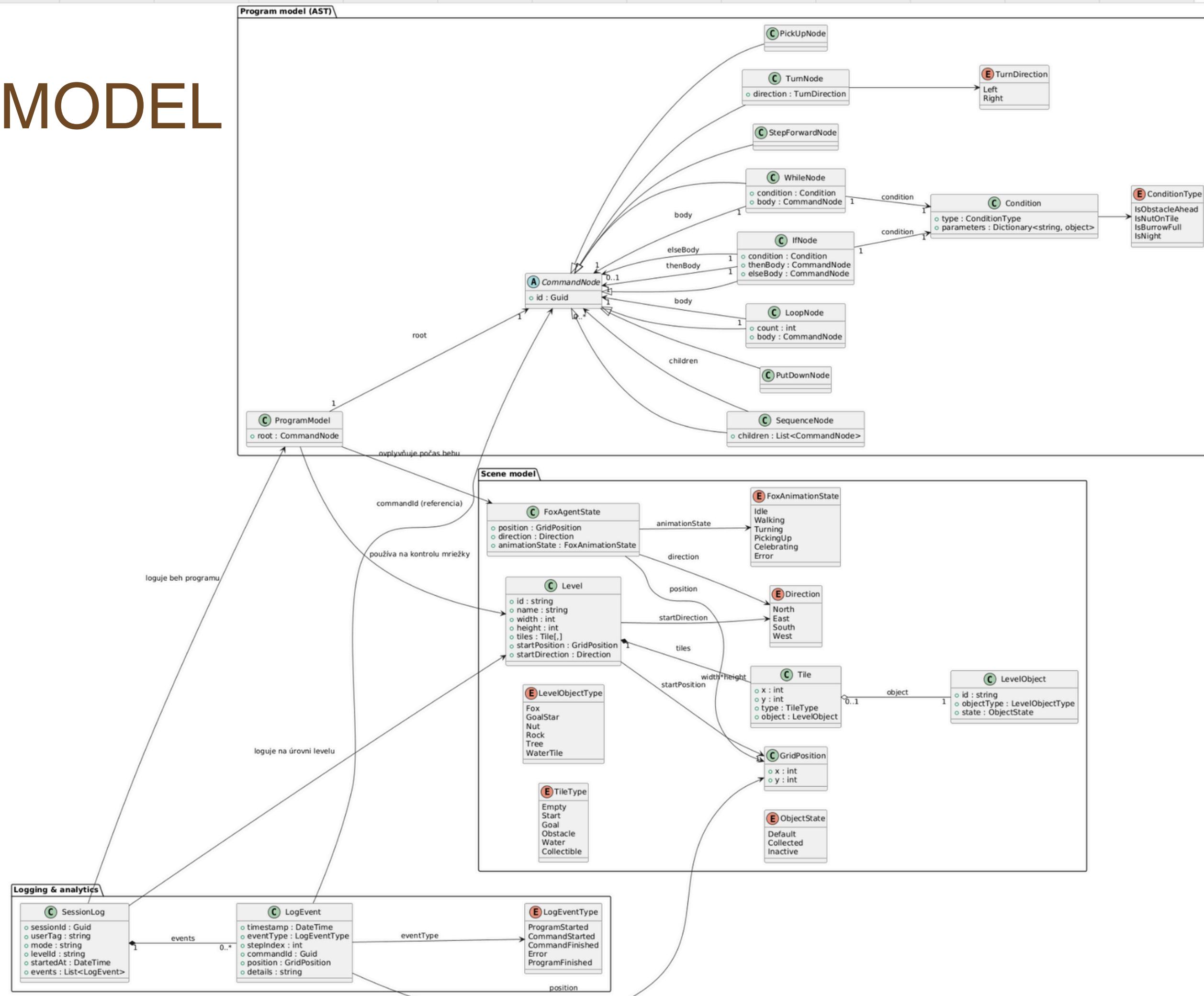
Kľúčové dátové štruktúry

- 2D mriežka Tile[x,y]
- Agent (pozícia, smer, stav)
- AST: ActionNode, LoopNode, IfNode, WhileNode

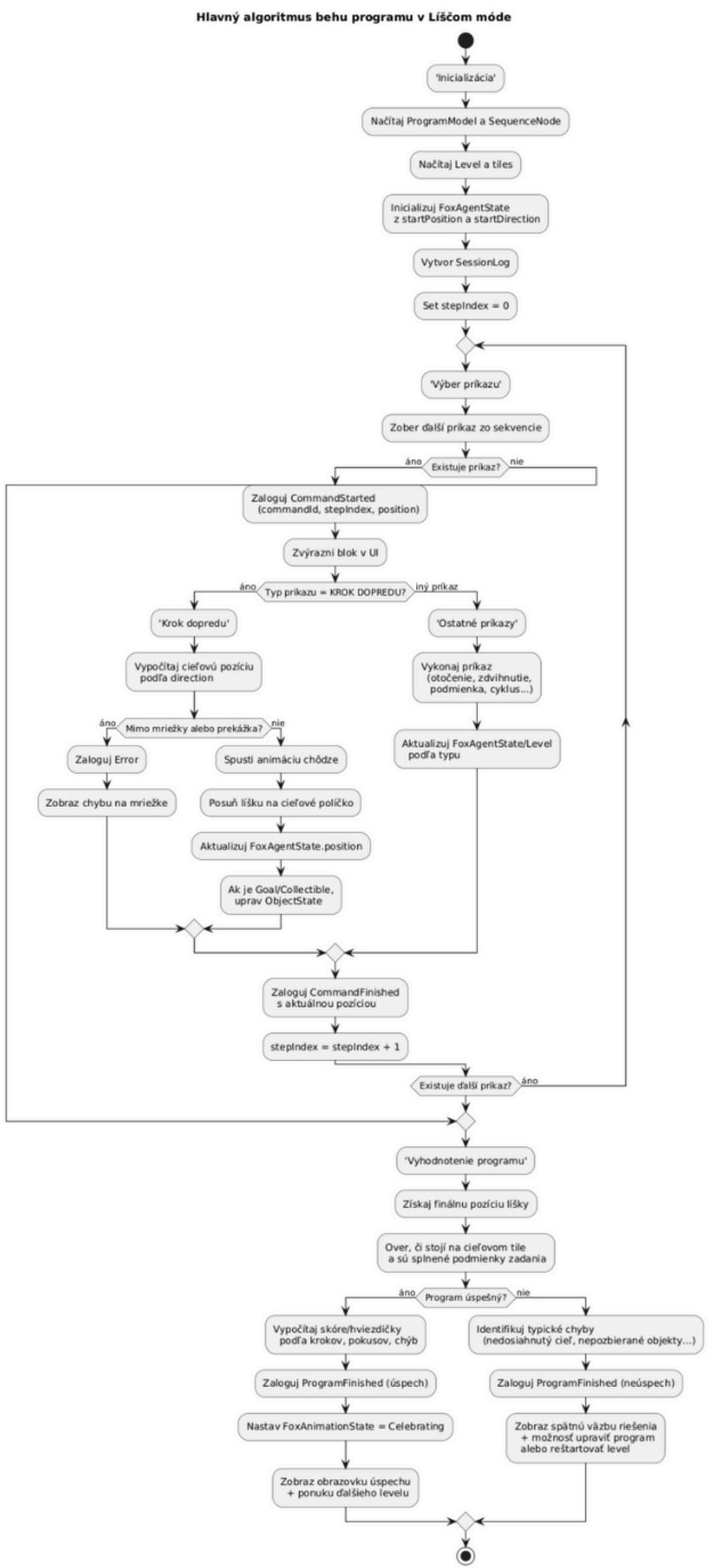
DIAGRAM PRÍPADOV POUŽITIA



DÁTOVÝ MODEL



ALGORITMUS LÍŠČÍ MÓD



LÍŠKA: SEKVENCIE (CESTA CEZ LES)

Koncept

- Prostredie na tvorbu lineárnych sekvencií príkazov.
- Analýza krokov, chýb a interakčných vzorcov pri skladaní programu.

Prostredie

- Mriežková mapa s objektmi (prekážky, ciele, akčné prvky).
- Agent vykonáva príkazy v pevnom poradí.

Mechaniky

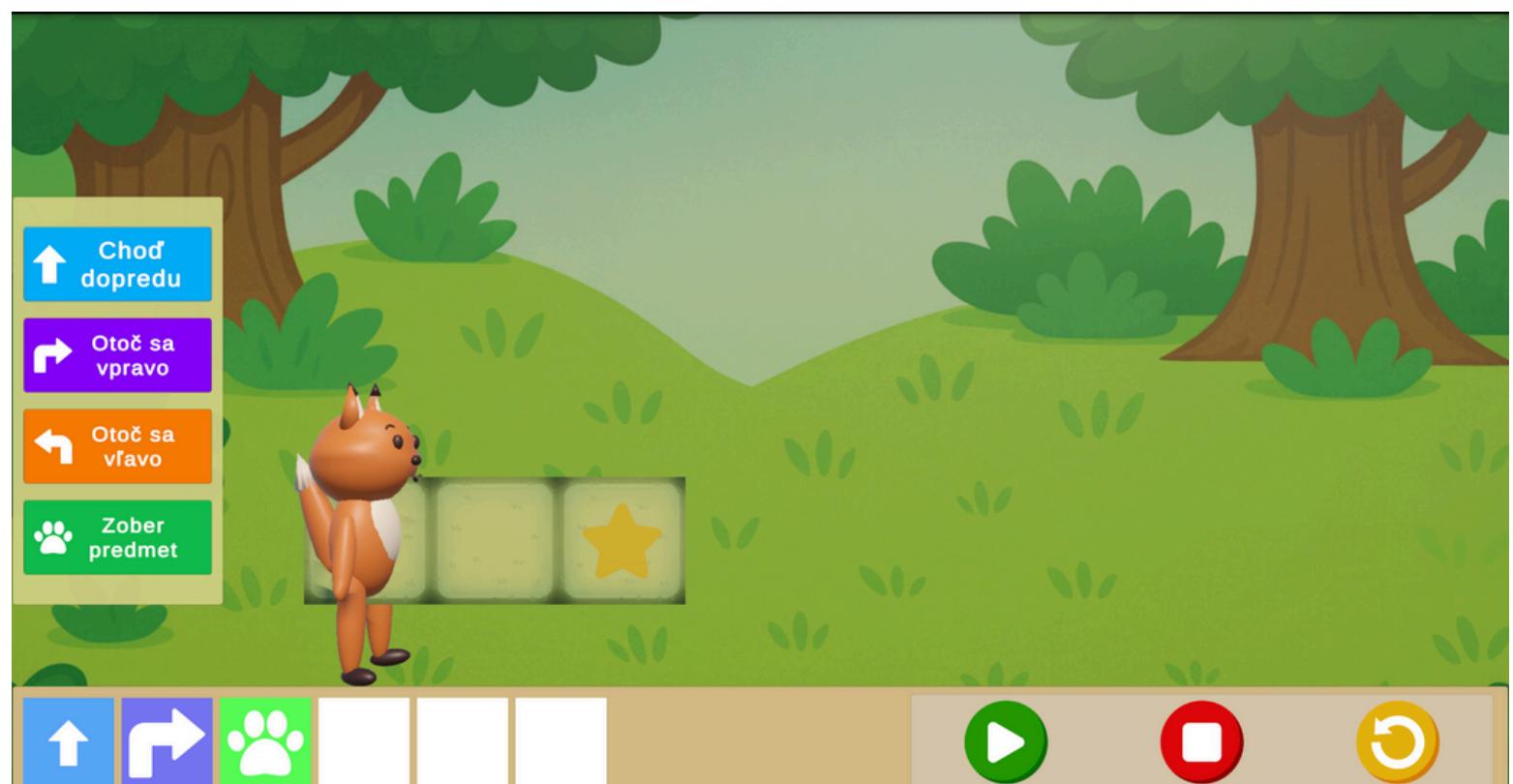
- Drag & drop blokov (pohyb, otočenie, akcia).
- Spustenie, zastavenie, krokovanie.
- Interpreter simuluje prechod medzi dlaždicami + loguje kroky.

Stupeň obtiažnosti:

- jednoduché krátke trasy → odbočky → prekážky → práca s predmetmi
- postupné rozširovanie palety blokov

Spätná väzba a hodnotenie:

- zvýrazňovanie aktuálne vykonávaného bloku
- jasné označenie miesta chyby pri neúspechu
- hviezdičkové hodnotenie efektívnosti
- logovanie: dĺžka programu, počet pokusov, typické chyby



ZAJAC: PODMIENKY (ZAJAČIA ZÁHRADKA)

Koncept

- Systém vizuálneho rozhodovania pomocou if–else štruktúr.
- Reakcia na dynamické stavy prostredia.

Vizualizácia rozhodovania:

- Zvýrazňovanie vetvy, ktorú interpreter vybral („pravda“ / „nepravda“)
- V krokovacom režime sa zobrazuje aktivovaná vetva pri každom kroku

Prostredie

- Scéna s premennými stavmi (dážď, tma, prekážky, objekty).
- Agent vyberá vetvu podľa aktuálneho stavu.

Stupeň obtiažnosti:

- jednoduchá binárna podmienka → kombinované stavy

Herné mechaniky:

- Bloky typu: ak <podmienka> potom {...} (+ voliteľný inak)
- Podmienky vychádzajú z vizuálne prezentovaných stavov (ikony dažďa, krtka, tmy)
- Telo vetvy obsahuje akčné bloky: preskoč, zalej záhon, schovaj sa

Spätná väzba a logovanie:

- počet aktivácií vetiev, zbytočné testy, chybné rozhodnutia
- hodnotenie založené na správnom návrhu vetvenia

VEVERIČKA: CYKLY (ORECHOVÉ DOBRODRUŽSTVO)

Koncept

- Práca s iteráciami: opakuj n-krát a opakuj kým .
- Optimalizácia opakovanych činností.

Prostredie

- Mriežka s viacerými objektmi (orechy, skrýše).
- Prirodzene opakujúce sa scenáre: zber, prenášanie, trasa.

Herné mechaniky:

- Bloky cyklov, do ktorých dieťa vkladá pohybové a akčné príkazy
- Pri „opakuj nx“ sa počet iterácií nastavuje číselníkom v bloku
- Pri „opakuj kým“ sú stavy v scéne vizualizované (napr. ukazovateľ plnosti skrýše)

Vizualizácia iterácií:

- odpočítavanie cyklu pomocou grafických „korálok“
- zvýraznenie tela cyklu počas opakovania

Stupeň obtiažnosti:

- pripravené cykly → dieťa vkladá obsah → dieťa tvorí celý cyklus
- neskôr aj vnorené cykly alebo kombinácia cyklus + podmienka

Spätná väzba a logovanie:

- počet iterácií, efektívnosť riešenia („úspora krokov“)
- detekcia nekonečných slučiek (časový limit / limit krokov)
- identifikácia mechanického používania cyklov bez pochopenia

SOVA & MEDVEĎ (MIX KONCEPTOV, REFLEXIA, SANDBOX)

SOVA – SOVIČKINE VÝZVY

Koncept

- Kombinácia sekvencií, podmienok a cyklov.
- Analýza optimalizačných stratégií používateľa.

Charakter úloh

- Premenlivé prostredie (noc/deň, prekážky).
- Časové alebo krokové limity.
- Viacero validných riešení.

Spätná väzba:

- prehľad použitých blokov, aktivovaných vетiev, počtu iterácií
- možnosť prehrania riešenia (replay)
- postupné nápovedy (najprv náznak → až potom konkrétny tip)

MEDVEĎ – BRLOH NÁPADOV (SANDBOX)

Koncept

- Otvorené prostredie pre tvorbu vlastných programov a scén.
- Plná paleta blokov, voľná manipulácia s objektmi.

Funkcie

- Pridávanie objektov do scény.
- Programovanie správania objektov.
- Ukladanie a načítanie projektov.

Sledovanie progresu:

- dĺžka programu, diverzita blokov, počet úprav
- kontrola technickej správnosti (zacyklenie, neplatné stavy)

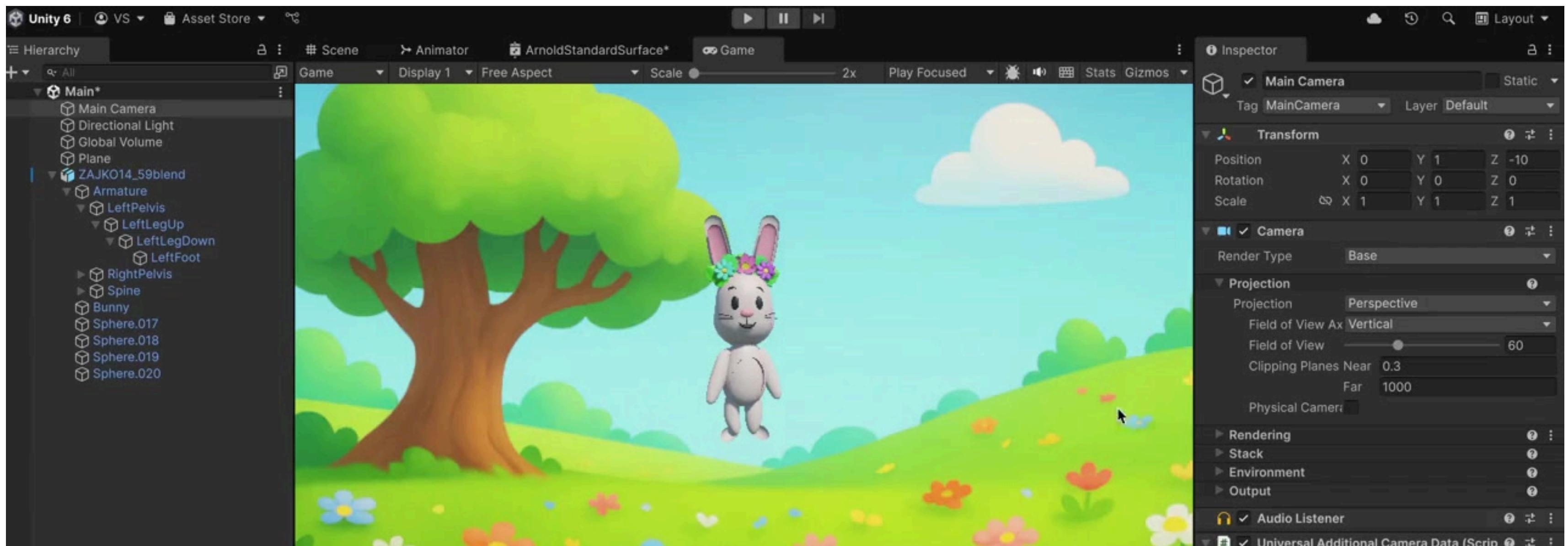
POSTAVY V BLENDER

Ukážka namodelovaných postavičiek v Blender na základe dokumentácie pripravených na implementáciu v Unity editore



UKÁŽKA IMPLEMENTÁCIE

Príprava na základe Unity dokumentácie



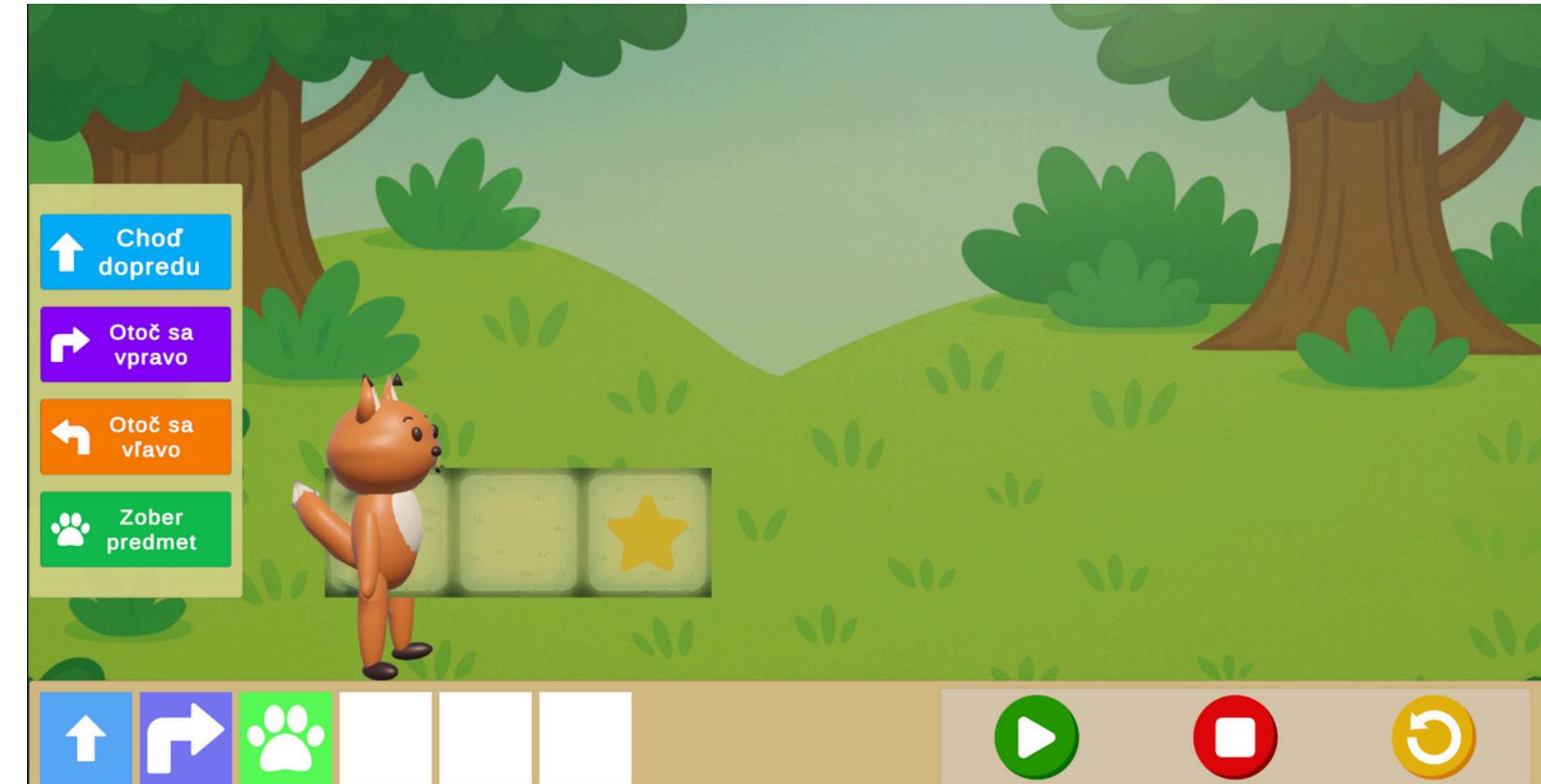
TESTOVANIE LÍŠČIEHO MÓDU V TRIEDE

Testovacia skupina:

- 13 žiakov vo veku 6–10 rokov
- Aktivita prebiehala v triede na tabletoch
- Použitá bola verzia rozhrania iba s ikonami (bez textu)

Priebeh:

- Deti mali samostatne prejšť úlohy Líščieho módu
- Prebiehalo priame pozorovanie ich práce a správania
- Po skončení sme sa žiakov pýtali na ich skúsenosť s ovládaním



Zistenia:

- Deti, ktoré ešte nevedeli plynulo čítať, sa orientovali podľa ikon bez problémov
- Pre čitateľov (najmä 8–10 rokov) bol sprievodný text nápomocný
- Text im pomohol rýchlejšie pochopiť význam ikon a rozhodovať sa pri úlohách

Dôsledok pre návrh:

- Do rozhrania sme doplnili kombináciu ikon + krátkeho sprievodného textu, aby bolo vhodné pre predčitateľov aj čitateľov.

FESSAKIS ET AL. (2013): PROBLEM SOLVING IN LOGO-BASED ENVIRONMENT

Zameranie článku:

- Prípadová štúdia o riešení problémov deťmi 5-6 r. v prostredí založenom na Logo.

Hlavné výsledky článku:

- Programovanie podporovalo rozvoj matematických konceptov, riešenie problémov a sociálne interakcie.
- Deti používali rôzne stratégie riešenia úloh, vrátane plánovania, pokus-omyl a spolupráce.
- Výskum ukázal, že programovanie bolo pre deti atraktívne a motivujúce.
- Podporované boli aj učiteľsky riadené a kolektívne aktivity.
- Štúdia potvrzuje realizovateľnosť integrácie programovania do bežných aktivít materskej školy.

WEINTROP & WILENSKY (2019): TRANSITION FROM BLOCK-BASED TO TEXT-BASED PROGRAMMING

Zameranie článku:

- Experimentálna štúdia o prechode študentov z blokového na textové programovanie.
- Hodnotenie konceptuálneho učenia, postojov a zmien v programátorských praktikách.

Hlavné zistenia článku:

- Po 5 týždňoch učenia existovali rozdiely medzi skupinou používajúcou blokové vs. textové prostredie, no po 10 týždňoch v Java rozdiely v znalostiach zmizli.
- Nebol zistený rozdiel v programovacích praktikách medzi skupinami po prechode na Java.
- Rozdiely správania medzi skupinami sa taktiež eliminovali po 15 týždňoch.
- Výsledky podporujú tvrdenie, že blokové prostredia môžu slúžiť ako účinné úvodné nástroje, no dlhodobý výkon sa vyrovná pri prechode na profesionálne jazyky.
- Štúdia rozširuje poznatky o tom, ako sa koncepty a postoje prenášajú medzi programovacími spôsobmi.

WANG & HANNAFIN (2005) - DBR PRE TECHNOLOGICKY PODPOROVANÉ UČENIE

Zameranie článku:

- Aplikuje DBR na vývoj technologických vzdelávacích systémov.

Kľúčové body:

- DBR slúži ako metodika pre tvorbu adaptívnych, interaktívnych softvériov.
- Dôraz na štyri dimenzie: teória → dizajn → implementácia → hodnotenie.
- Výskumník a dizajnér musia byť v neustálej interakcii.
- Technologické prostredia musia byť iteratívne, flexibilné a zakotvené v správaní používateľov.

ZOZNAM POUŽITEJ LITERATÚRY

zhodný so zoznamom literatúry v diplomovej práci