Softvér vo vzdelávaní

Bludisko so šípkami

Lukáš Gajdošech

7. mája 2024

Úvod

- Predstavenie hry "Bludisko so šípkami"
- Cieľ: Navigovať robota cez bludisko pomocou šípok
- Výukový nástroj pre stredné školy

Technológie

- JavaScript a knižnica p5.js pre interaktívnu grafiku
- Možné rozšírenia: File API pre načítanie máp z textových súborov
- Algoritmy pre kontrolu riešiteľnosti bludiska (napr. BFS alebo DFS)

Prečo p5.js?

- Ľahké spustenie na iných zariadeniach
- Zamerané na kreatívne kódovanie a vizualizácie
- Rýchle prototypovanie bez potreby predchádzajúcich webových znalostí
- Vstavané nástroje pre grafiku a animácie

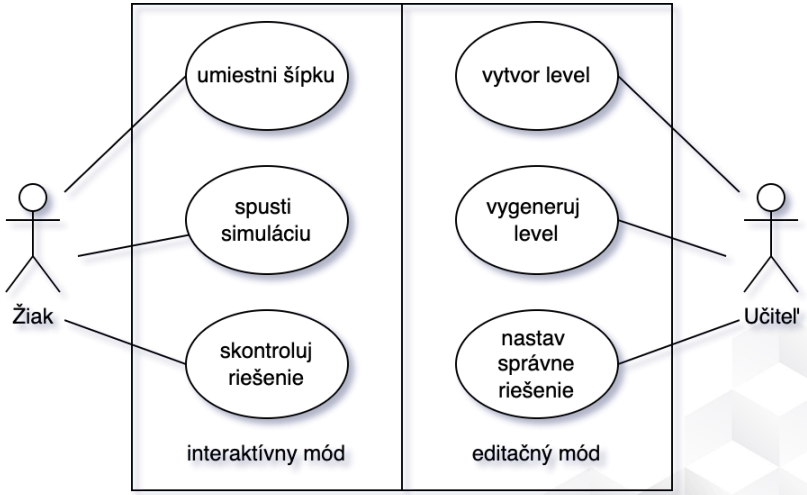
Základná štruktúra kódu

```
class Cell {
    // Vlastnosti a metóda pre zobrazenie bunky plochy
}

class Robot {
    // Pohyb a zobrazenie robota
}

class Arrow {
    // Umiestnenie a zobrazenie šípok
}
```

Diagram Použitia



Generovanie a načítanie máp

- Automatické generovanie máp s rôznymi rozmermi a prekážkami
- Načítanie máp z textového súboru pre personalizované úrovne
- Výzva pre študentov: Navrhni svoje vlastné bludisko

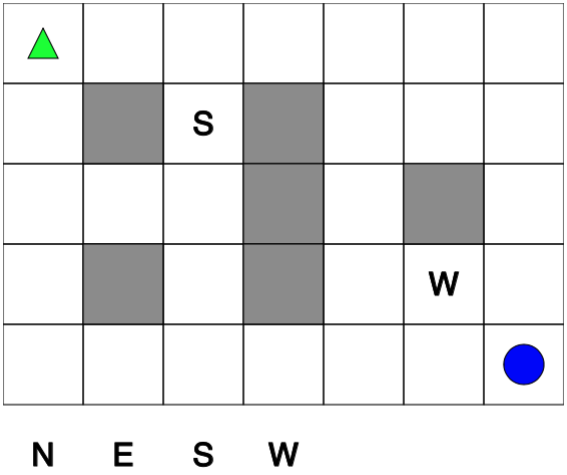
Využitie vo vyučovaní

- Rozvoj problémového myslenia a základov programovania
- Práca s grafmi a algoritmami na hľadanie ciest
- Tímová práca a dizajn vlastných úrovní
- Diskusia o riešeniach a stratégiách

Nápady na rozšírenie

- Viacúrovňové bludiská
- Dynamické prekážky a nepriatelia
- Editor máp pre užívateľov
- Programovacie výzvy
- Integrácia s učebnými platformami

Prototyp



Záver

- "Bludisko so šípkami" poskytuje zábavnú a edukatívnu platformu
- Podporuje logické myslenie, programovanie a matematické zručnosti
- Rozširuje možnosti vyučovania s praktickými aplikáciami

Prískok 1

19.3.2024

Žiacky Režim

- Umiestňovanie šípok
- Spustenie simulácie
- Kontrola riešenia
- Premazanie konfigurácie
- Načítanie mapy zo súboru
- `https://github.com/gajdosech2/EduSoft`

Štruktúra súborov

```
sipkyApp
├── main.html
├── p5.js
├── map0.txt map0.txt map1.txt ...
├── savedSolutions
│   └── student0.txt student1.txt ...
└── source
    └── robot.js arrow.js cell.js ...
```

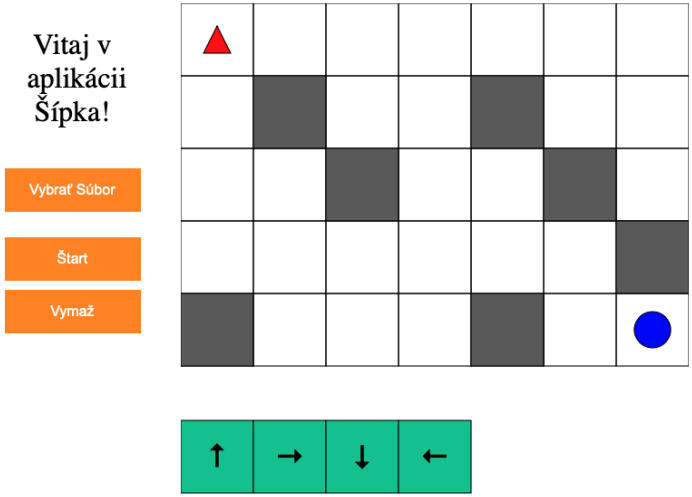
Pohyb robota

```
step() {
    let arrow = arrows.find(a => a.x === this.x && a.y === this.y);
    if (arrow) this.dir = arrow.direction;
    switch (this.dir)
        case "↑":
            if (this.y > 0 && !grid[this.y - 1][this.x].wall)
                this.y--;
            else {this.dir = "→"; this.step();}
            break;
        case "→":
            if (this.x < cols - 1 && !grid[this.y][this.x + 1].wall)
                this.x++;
            else {this.dir = "↓"; this.step();}
            break;
        ...
}
```

Animácia pohybu

```
move() {
    this.check();
    if (!this.moving) return;
    this.step();
    setTimeout(() => this.move(), STEP_TIME_MS = 500);
}
```

Demo



Nabudúce

- Náhodné generovanie máp ✓
- Uloženie riešenia do súboru ✓
- Trojuholník reaguje na smer ✓
- Tlačidlá menia text (pauza) ✓
- Umiestnené šípky sa dajú premiestniť ✓
- Automatické hľadanie riešenia ✓
  - po 2 neúspechoch
- Upozornenie na lepšie riešenie ✓

Prískok 2

Z minula

- Náhodné generovanie máp ✓
- Uloženie riešenia do súboru ✓
- Trojuholník reaguje na smer ✓
- Tlačidlá menia text (pauza) ✓
- Umiestnené šípky sa dajú premiestniť ✓
- Automatické hľadanie riešenia ✓
  - po 2 neúspechoch
- Upozornenie na lepšie riešenie ✓

Náhodná mapa

```
function createRandomGrid() {
    const randomColumns = Math.floor(Math.random() * (9 - 4 + 1)) + 4;
    const randomRows = Math.floor(Math.random() * (7 - 4 + 1)) + 4;
    const wall = (Math.floor(Math.random() * (15 - 5 + 1)) + 5) / 100;
    const endX = Math.floor(Math.random() * (randomColumns - 1)) + 1;
    const endY = Math.floor(Math.random() * (randomRows - 1)) + 1;
    let gridString = "";
    for (let i = 0; i < randomRows; i++) {
        for (let j = 0; j < randomColumns; j++) {
            if (Math.random() < wall) gridString += '#';
            else gridString += '.';
        }
    }

    return gridRows.join('\n');
}
```

Brute-force Kontrola splniteľnosti

```
function bruteForceSolve() {
    maze = convertToStringArray();
    foundSolution = false;
    bestSolution = null;
    bestSteps = Infinity;

    for (let a = 0; a < MAX_ARROWS; a++) {
        if (foundSolution) break;
        arrowsRecursion(maze.map(row => [...row]), a);
    }

    if (!foundSolution) alert("Neriešiteľné bludisko!");
    else placeArrows(bestSolution);
}
```

Nastavenie a Paralelizácia

```
const EARLY_STOP = false;
const MAX_STEPS = 30;
const MAX_ARROWS = 4;
```

Hľadanie riešenia je implementované ako JavaScript Worker,
vďaka čomu nespôsobuje zamrznutie hlavného vlákna programu.

Nabudúce

- Režim na tvorbu mapy
- Teleporty?
- Zlepšenie generovanie máp
- Optimalizácia hľadania riešenia

Prískok 3

6.5.2024

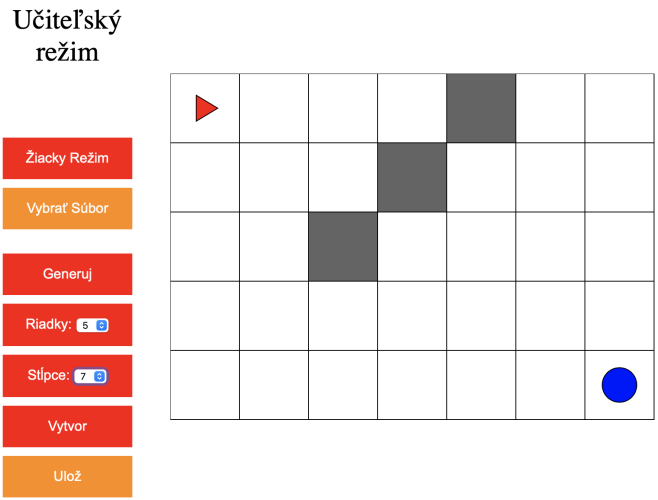
Novinky

- Režim na tvorbu mapy ✓
- Prepínanie režimov ✓
- ~~Teleporty?~~
- Zlepšenie generovanie máp ✓
- Optimalizácia hľadania riešenia ✗

Prepínanie režimov

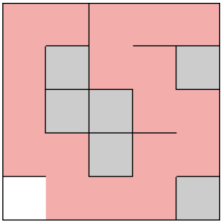
```
function switchMode() {
    maze = convertToStringArray().map(row => row.join('')).join('\n');
    sessionStorage.setItem('mazeData', maze);
    window.location.href = 'main.html';
}
```

Učiteľský režim

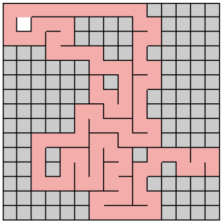


Generovanie Labyrintu

The recursive backtracker was my go-to algorithm for years. It's fast, easy to implement, and generates mazes that are (to my eyes, at least) quite esthetically pleasing. Of all the algorithms, it generates the fewest dead-ends, and as a result has particularly long and winding passages. It's especially hypnotic to watch the algorithm in action!



Reset Step Watch Run



Reset Step Watch Run

Buck, Jamis. "Maze Generation Algorithm Recap." Jamis Buck's Blog, 7 Feb. 2011, `https://weblog.jamisbuck.org/2011/2/7/maze-generation-algorithm-recap`

Pozícia konca

```
// Vážený výber pre koncovú pozíciu E viac vpravo dole
let weightedEndPositions = [];
for (let y = 0; y < rows; y++) {
    for (let x = 0; x < cols; x++) {
        if (maze[y][x] === '.') {
            const weight = (x * y) * (x * y);
            for (let i = 0; i < weight; i++) {
                weightedEndPositions.push({x, y});
            }
        }
    }
}
const endPosition = weightedEndPositions[
Math.floor(Math.random() * weightedEndPositions.length)];
maze[endPosition.y][endPosition.x] = 'E';
```

Nabudúce

- Návod
- Optimalizácia hľadania riešenia ?

Priestor na vaše otázky...

Ďakujem za pozornosť!