Spy vs. Spy

Zápočtový program LS 2018/2019

Kristína Szabová

# Popis programu

Program je variácia známej hry Spy vs. Spy z roku 1984. Hlavné postavy sú dvaja špióni, biely a čierny. Dej sa odohráva na veľvyslanectve, ktoré má rôzne poprepájané miestnosti s rôznymi druhmi nábytku. V niektorých objektoch sú schované predmety: pas, peniaze, kľúč a tajné plány. Cieľom špiónov je získať všetky tieto predmety a dostať sa cez jedny z dverí na letisko, odkiaľ ich môže bezpečne dopraviť na svoju základňu. Zároveň sa musí snažiť zabrániť svojmu protivníkovi, aby sa mu to podarilo prvému. To je možné rôznymi spôsobmi: nastražením pasce, bojom s protivníkom atď. Spy vs. Spy je teda hra, v ktorej záleží na stratégií.

V tejto verzii hry hrá za bieleho špióna hráč a za čierneho počítač. Počet levelov je 3, celkovo vyhrá ten hráč, ktorý ich vyhral väčší počet.

# Pravidlá a ovládanie hry

Najdôležitejšou časťou hry je zbieranie predmetov. Predmety sú uložené v nábytku jednotlivo, jeden z nich je v kufríku. Hráč môže naraz držať iba jeden predmet, pokiaľ nemá kufrík. Získať nejaký z predmetov je možné zdvihnutím nábytku, v ktorom je uložený, alebo bojom s hráčom, ktorý ho práve drží.

Hráč sa môže pohybovať po všetkých miestnostiach v rámci vyhradenej plochy – "podlahy". Do inej miestnosti prechádza cez dvere v prípade, že sú otvorené. Otvárať a zatvárať ich môže priamo hráč.

Pokiaľ sa obidvaja hráči nachádzajú v jednej miestnosti a v určitej vzdialenosti, môže sa jeden z nich rozhodnúť s druhým bojovať. Na zabitie protihráča je potrebných 10 zásahov. Počítadlo sa obnovuje po oživení a po prejdení do inej miestnosti.

Na každý level je vyhradený čas, počas ktorého musia hráči nájsť všetky predmety a dostať sa na letisko. Počiatočný čas je spoločný pre oboch hráčov, avšak obidvaja majú vlastné počítadlo. Smrťou sa hráčovi čas zníži o 15 sekúnd, k čomu treba prirátať približne 7 sekúnd predtým, ako znova ožije. Môže sa teda stať, že jednému z hráčov vyprší časomiera skôr ako druhému.

V prípade smrti hráča sa všetky predmety, ktoré práve držal, buď priradia protivníkovi, ak ho zabil on, alebo uložia do náhodného prázdneho nábytku v miestnosti.

Hráč sa môže pokúsiť zabrániť protivníkovi v nájdení predmetov nastražením pascí. Typy pascí sú tri: časová bomba, bomba a vedro s vodou. Líšia sa použitím aj možnosťami obrany pred nimi. Časová bomba sa nastavuje do miestnosti. Po aktivovaní sa začne 30-sekundový odpočet, na konci ktorého zomrú všetci hráči, ktorí sa v miestnosti práve nachádzajú. Pred časovou bombou sa nedá brániť, je potrebné si zapamätať, v ktorej miestnosti sa nachádza. Obyčajná bomba sa nastavuje do ľubovoľného nábytku. Aktivuje sa jeho zodvihnutím a nezáleží pritom, či je to hráč, ktorý pascu nastražil. Chrániť pred bombou sa dá pomocou štítu, ktorý hráč nájde v ľubovoľnej skrinke s prvou pomocou. To, či práve má štít, spozná podľa toho, že obrázok hráča sa zmení na štít. Poslednou pascou je vedro s vodou, ktoré sa nastražuje na dvere. Aktivuje sa prejdením cez tieto dvere. Ochrana pred vedrom je dáždnik, ktorý nájdeme na každom vešiaku. To, či ho hráč práve má, sa opäť spozná tým, že jeho obrázok sa zmení na dáždnik.

Pascu nie je možné nastražiť, ak sa v miestnosti nachádzajú obidvaja hráči.

Hráč ovláda svojho špióna prostredníctvom klávesnice. Jeho pohyb sa určuje šípkami alebo klávesami W, A, S, D. Otvárať aj zatvárať dvere je možné klávesou X. Rovnako klávesou X môžeme nadvihnúť nábytok. Ak chceme bojovať proti protihráčovi, používame medzerník. Tieto funkcie ale fungujú iba vtedy, ak sa hráč nachádza v dostatočnej blízkosti dverí, nábytku alebo protihráča. Pre nastraženie pasce musíme najskôr aktivovať Trapulátor, čo spravíme klávesou Z. Následne je potrebné vybrať si druh pasce: 1 znamená časová bomba, 2 vedro s vodou a 3 bomba. Následne sa aktivuje pomocou X. V prípade vedra a bomby je potrebné, aby sa hráč v momente aktivovania nachádzal v blízkosti dverí alebo nábytku, inak sa pasca stratí. Nakoniec, klávesou H môžeme zobrazovať a schovávať prehľad ovládania.

# Formát levelov

Každý level má určitý počet miestností, ktorý môže byť rôzny. To, že miestnosti majú spoločnú stenu, nemusí znamenať, že sa medzi nimi dá prejsť. Všeobecne sa medzi miestnosťami dá prechádzať iba vtedy, keď sú medzi nimi dvere. Existujú štyri druhy dverí, jedny na každej stene miestnosti, vrátane tej "neviditeľnej", teda tej, z ktorej sa pozerá hráč.

V miestnosti sa ďalej môže nachádzať ľubovoľný počet nábytku. Nábytok existuje v ôsmich typoch: polička na knihy, písací stôl, malá polička, polička, mikrovlnka, komoda, vešiak na kabáty a skrinka na prvú pomoc. Všetky okrem posledných dvoch môžu obsahovať nejaký predmet, vešiak a skrinka obsahujú dáždnik alebo štít na ochranu pred pascami.

Pri vytváraní mapy levelu platia tieto zásady:

* Ak z miestnosti A vedú dvere do miestnosti B, potom aj z miestnosti B vedú dvere do miestnosti A. Navyše, dvere sa musia priestorovo zhodovať, t.j. napríklad ak z miestnosti A vedú do B dvere napravo, potom z miestnosti B do miestnosti A vedú dvere naľavo.
* Všetky dvere musia viesť do nejakej miestnosti, s výmkou dverí, ktoré vedú na letisko.
* V jednej miestnosti sa môže nachádzať každý nábytok iba raz.
* Kvôli vzájomnej polohe sa v jednej miestnosti nesmú naraz vyskytovať nasledovné dvojice nábytku a dverí: písací stôl a malá polička, písací stôl a vešiak na kabáty, malá polička a vešiak na kabáty, polička a mikrovlnka, komoda a skrinka prvej pomoci, písací stôl a dvere na zadnej stene, písací stôl a polička.

Mapy levelov sú uložené v priečinku "./Assets/LevelMaps/". Súbory sú pomenované "levelX.txt", kde X je číslo levelu. (Prvý level má číslo 1.) Formát súborov je nasledujúci:

Na prvom riadku sú 3 čísla oddelené medzerou:

počet\_poschodí počet\_riadkov počet\_stĺpcov

Ďalej nasleduje počet\_poschodí\*počet\_riadkov\*počet\_stĺpcov skupín riadkov v nasledujúcom formáte:

farba\_miestnosti

počet\_nábytku

typ\_nábytku1 predmet\_v\_nábytku1 typ\_nábytku2 predmet\_v\_nábytku2 ...

počet\_dverí

typ\_dverí1 poschodie,riadok,stĺpec[miestnosť, kam vedú dvere 1] typ\_dverí2 poschodie,riadok,stĺpec ...

Na posledných dvoch riadkoch sa vo formáte poschodie,riadok,stĺpec nachádzajú súradnice miestností, kde sa na začiatku objavia (1) hráč a (2) počítač.

# Komponenty programu

Program pozostáva z viacero classes, z ktorých každá má na starosti riadenie nejakej časti programu.

## class Game

Class Game je hlavnou časťou hry. Stará sa o inicializáciu hry, ukončenie, sleduje jej stav a riadi hráčov. Všetky jej metódy aj atribúty sú statické. **Atribúty** sú nasledovné:

*int NUMBER\_OF\_LEVELS* – určuje počet levelov, ktoré má hra celkovo

*bool started* – true, ak hra už začala

*bool stopped* – true, ak je hra zastavená (napr. pri načítavaní nového levelu)

*int current\_level*

*int computer\_wins* – počet levelov, ktoré doteraz vyhral počítač

*int human\_wins* – počet levelov, ktoré doteraz vyhral človek

*Room[,,] levelMap* – mapa levelu, viď popis class Room ďalej

*Player[] players* – pole hráčov, players[0] je človek, players[1] je počítač

*Room[] rooms* – miestnosti aktuálne načítané v hornom (0) a dolnom (1) paneli, viď class UI

*Room noRoom* – špeciálna miestnosť, ktorá slúži ako placeholder pre neexistujúce miestnosti

*Room airport* – špeciálna miestnosť, do ktorej sa hráč dostane po pozbieraní všetkých predmetov

**Metódy:**

*void EventOnKeyPress(char key*) – popisuje, aká udalosť sa má stať po stlačení klávesy

*void LoadRoomByDoor(int door, int player)* – na základe toho, ktorý hráč (player) prešiel cez daný typ *dverí (door),* načíta hráčovho panelu miestnosť, do ktorej tieto dvere vedú

*void LoadAirport(int player)* – podľa toho, ktorý hráč sa snaží dostať na letisko, buď ho načíta a prípadne začne ďalší level, alebo odmietne

*void Stop(int loser)* – zastaví hru a zobrazí, ktorý hráč vyhral hru

*void InitializeNextLevel()* – načíta mapu levelu, vynuluje počítadlá a zobrazí hráčov v ich miestnostiach

*void InitializeGame(Form1 parentForm)* – načíta základné časti UI a určí, kde sa majú vykresliť (parentForm)

## class Player

Kontroluje správanie hráča, sleduje stav.

**Atribúty:**

*int playerType* – určuje, či je hráč človek (0), alebo počítač (1)

*int state* – určuje, či sa hráč nachádza v Trapulátore (1), drží pascu (2), alebo nič z toho (0)

*PictureBox playerImage* – Control, v ktorom sa vykresľuje hráč

*Position playerPosition* – pozícia hráča, viď class Position

*Coordinates playerImageCoordinates –* súradnice, kde sa má vykresliť obrázok hráča (pozor, sú iné ako samotné súradnice hráča, pretože tie určujú polohu jeho nôh)

*Size imageSize* – veľkosť obrázku hráča

*int panelOnScreen* – v ktorom paneli sa nachádza hráč

*int trap* – pasca, ktorú drží hráč

*int disarm* – obrana, ktorú drží hráč

*int secondsLeft* – počítadlo zostávajúceho času

*int numberOfItems* – počet predmetov, ktoré práve má

*int health* – počítadlo, koľko zásahov ešte môže zvládnuť, kým umrie

*bool[] items* – pre každý predmet určuje, či ho má (true), alebo nie (false)

*string aliveImage, deadImage, umbrellaImage, shieldImage, fightingImage,* suitcaseImage – obrázky, ktoré sa vykresľujú pri rôznych stavoch

**Metódy**:

*void MovePlayer(char direction)* – pohne hráča do daného smeru, ak je to možné

*void StopDoingActions()* – zabráni hráčovi pohybovať sa, bojovať atď.

*void Hit(int opponent)* – zasiahne protihráča, ak je pri ňom

*int PickUpItem(int item)* – nastaví predmet (item), že ho vlastní; vráti hodnotu predmetu, ktorý sa uloží do nábytku, ktorý zodvihol

*void DropItemToFurniture(int furniture*) – vloží všetky predmety, ktoré má, do daného nábytku

*void LoseAllItems()* – nastaví všetky predmety v items na false

*void PickUpDisarm(int furniture)* – zoberie obranu z daného nábytku

*void DropDisarm()* – stratí obranu, ktorú mal

*void SwitchPanel()* – prekreslí hráča do opačného panelu na obrazovke

*void DisplayPlayerInView()* – nakreslí hráča do svojho panelu

*void UpdateImageInNewRoom(int doorCrossed)* – nakreslí hráča do druhej miestnosti na správne miesto

*void RemovePlayerFromView()* – skryje hráča

*void UpdatePlayerImageCoordinates()* – vypočíta nové súradnice obrázku hráča podľa jeho polohy

*void SetDoorBeingCrossed(Coordinates pos, ref int doorCrossed)* – zistí, či hráč prechádza cez dvere, ak áno, nastaví hodnotu doorCrossed na číslo týchto dverí

*bool ValidateDoorCrossing(char direction, int door)* – vráti true, ak sa smer pohybu zhoduje s typom dverí

*Coordinates CalculatePositionAfterCrossingDoor(int door, Coordinates curCoordinates)* – vráti pozíciu po prejdení dverí

*int ItemInPosession()* – vráti číslo predmetu, ktorý hráč drží

*bool ArePlayersClose()* – vráti true, ak sa hráči nachádzajú vedľa seba

## class Room

Popisuje obsah miestnosti.

**Atribúty:**

*char color* – farba miestnosti

*int occupiedBy* – číslo hráča, ktorý sa práve nachádza v miestnosti; -1 ak žiadny, 2 ak obidvaja

*Furniture[] furnitures* – pole s nábytkom; furnitures[i] obsahuje objekt Furniture, ak sa i-tý nábytok v miestnosti nachádza, inak null

*Door[] doors* – pole s dverami; doors[i] obsahuje objekt Door, ak sa i-té dvere nachádzajú v miestnosti, inak null

*List<int> furnituresPresent* – čísla nábytku, ktorý miestnosť obsahuje

*List<int> doorsPresent* – čísla dverí, ktoré sa nachádzajú v miestnosti

*List<ImageContainer>* *images* – obrázky, ktoré sa majú zobraziť v miestnosti

**Metódy**:

*void LoadRoom(TransparentPanel frame, int player*) – načíta miestnosť do panelu (frame) a umiestni do nej hráča (player)

*void LeaveRoom(int player)* – nastaví atribút occupiedBy na takú hodnotu, aby sa zhodovala s tým, že hráč player už v miestnosti nie je

*void AddPlayer(int player) –* pridá do miestnosti hráča (player)

*int FurnitureNearby(Coordinates playerPosition)* – vráti číslo nábytku, pri ktorom sa nachádza hráč v miestnosti

*int DoorNearby(Coordinates playerPosition*) – vráti číslo dverí, pri ktorých sa nachádza hráč

*void AddFurniture(int type, int item)* – vloží do miestnosti daný typ nábytku

*void AddDoor(int type, Triplet leadsTo)* – vloží do miestnosti daný typ dverí

*int GetRandomEmptyFurniture() –* vráti náhodné číslo jedného z nábytkov v miestnosti, v ktorom nie je žiadny predmet

*void ActivateTimeBomb*() – v danej miestnosti aktivuje časovú bombu a po 10 sekundách zabije všetkých hráčov nachádzajúcich sa v miestnosti

*string RoomFilename()* – pomocou atribútu color vytvorí názov súboru s pozadím pre danú miestnosť

## class Furniture

Má na starosti správanie nábytku.

**Atribúty**:

*int type* – číslo nábytku od 0 do 7

*int item* – predmet, ktorý sa v danom nábytku nachádza

*bool trap* – true ak obsahuje pascu

*ImageContainer image* – obrázok, ktorý sa vykresľuje na obrazovke

*Size imageSize* – veľkosť obrázka

*Coordinates imagePosition* – súradnice, kde je obrázok umiestnený

*string filename* – názov súboru s obrázkom nábytku

**Metódy**:

*void Lift(int player)* – zdvihne nábytok a hráčovi (player) priradí predmet, ktorý sa v ňom nachádza

*void Release(int player)* – vráti nábytok na pôvodné miesto

*bool PositionInRangeOfFurniture(int furnitureType, Coordinates position)* – v závislosti na type nábytka vráti true, ak je position v určitej vzdialenosti od nábytku

*Coordinates GetCenterLocation(int furnitureType)* – vráti súradnice stredu daného nábytku

*void CalculateImagePosition()* – nastaví hodnotu atribútu imagePosition

*void CalculateImageSize()* – nastaví hodnotu atribútu imageSize

*void SetFilename()* – nastaví hodnotu atribútu filename

## class Door

Má na starosti správanie dverí.

**Atribúty**:

*int location* – typ dverí

*bool open* – true ak sú dvere otvorené

*Triplet leadsTo* – súradnice miestnosti, do ktorej dvere vedú

*bool trap* – true, ak je na dverách nastavená pasca

*ImageContainer image* – obrázok na vykreslenie dverí

*Size imageSize* – veľkosť obrázka

*Coordinates imageCoordinates* – súradnice, na ktoré sa má obrázok vykresliť

*string openFileName* – obrázok otvorených dverí

*string closedFileName* – obrázok zavretých dverí

**Metódy**:

*int Switch(int player)* – otvorí, resp. zatvorí dvere a vráti 0 ak sa to podarilo, inak sa pokúsi načítať letisko a vráti -1

*bool PositionInRangeOfDoor(int location, Coordinates position*) – vráti true, ak je pozícia dostatočne blízko na otvorenie dverí typu location

*bool PositionInDoor(int location, Coordinates position) –* vráti true, ak je pozícia dostatočne blízko na prejdenie dverí

*int GetCorrespondingDoor(int location)* – vráti location dverí, ktoré sú na druhej strane steny

*Coordinates GetCenterLocation(int location*) – vráti súradnice stredu dverí

*void Open(int player) –* otvorí dvere

*void Close(int player)* – zavrie dvere

*void CalculateImagePosition*() – nastaví hodnotu atribútu imagePosition

*void CalculateImageSize()* – nastaví hodnotu atribútu imageSize

*void SetFilename()* – nastaví hodnotu atribútov openFileName a ClosedFileName

## class Item

Zobrazuje jednotlivé predmety.

**Atribúty**:

*string placeholderImage* – zástupca obrázka, ktorý sa zobrazí, keď hráč nevlastní daný predmet

*Size imageSize* – veľkosť obrázka

**Metódy**:

*void InitializeItems()* – pridá na trapulátor placeholderImage pre každý predmet

*void ShowOnTrapulator(int item, int player)* – pridá daný predmet danému hráčovi

*void HideFromTrapulator(int item, int player)* – odoberie danému hráčovi predmet

*string GetFilename(int item)* – vráti názov súboru, v ktorom sa nachádza obrázok daného predmetu

*Coordinates CalculatePositionOnTrapulator(int item)* – vráti súradnice, na ktorých sa zobrazí obrázok predmetu

## class Suitcase

Pamätá si obsah kufríka.

**Atribúty**:

*bool[] contents* – true pre každý predmet, ktorý kufrík obsahuje

*int numberOfItems* – počet predmetov, ktoré sa v kufríku nachádzajú

**Metódy**:

*void AddItem(int item) –* pridá do kurfíka daný predmet

## class Trap

Nastavuje a aktivuje pasce.

**Metódy**:

*void SetTrap(Room r)* – nastaví časovú bombu do danej miestnosti

*void SetTrap(Furniture f) –* nastaví bombu do daného nábytku

*void SetTrap(Door d) –* nastaví vedro na dané dvere

*void Activate(int player, int trapType*) – zabije hráča, ak nemá obranu proti pasci

## class ComputerAI

Jednoduchá umelá inteligencia, ktorá sa stará o správanie hráča riadeného počítačom, viď ďalšia sekcia.

**Classes**:

*DoorsInRoom* – pamätá si navštívené a nenavštívené dvere v miestnosti pre účely hľadania letiska

*FurnitureAndDoorsInRoom* – pamätá si navštívené a nenavštívené dvere a nábytok pre účely hľadania predmetov

**Atribúty**:

*Player computer* – hráč riadený počítačom

*Player opponent* – hráč riadený človekom

*DoorsInRoom[,,] memoryDoors –* pamätá si dvere pre každú miestnosť na mape

*FurnitureAndDoorsInRoom[,,]* *memoryObjects* – pamätá si dvere a nábytok pre každú miestnosť

*List<Door>* *doorsToAirportStack* – zásobník prejdených dverí pri hľadaní letiska

*List<Door> doorsToExploreStack* – zásobník prejdených dverí pri hľadaní predmetov

*bool hitLastTime* – true ak pri poslednej akcii zasiahol hráča

**Metódy**:

*void Start()* – nastaví Timer pre akcie počítača

*void CreateMemory(Triplet levelShape)* – nastaví tvar atribútov memoryDoors a memoryObjects

*void CopyObjectsIntoRoom(Triplet roomCoords)* – nastaví všetky objekty v miestnosti ako nenavštívené v memoryDoors a memoryObjects

*char GetRandomDirection()* – vyberie náhodný smer pohybu

*void CalculateNextMove()* – na základe rozhodovacieho stromu (viď ďalšia sekcia) spraví pohyb

*void Fight()* – udrie protivníka

*void ExploreAllDoors(List<Door> doorsStack)* – prezrie zostávajúce dvere v miestnosti, ak nejaké ostávajú, inak sa vráti poslednými prejdenými dverami na doorsStack

*void ExploreLevel()* – prezrie neprezretý nábytok, ak v miestnosti nejaký zostáva, inak zavolá ExploreAllDoors

*void GoToLocation(Coordinates coords)* – pohne hráča smerom k daným súradniciam

## class Position, class Coordinates, class Triplet

Pomocné dátové štruktúry. Position ukladá poschodie, súradnice miestnosti na mape a súradnice v miestnosti. Coordinates ukladá dve čísla typu int. Triplet ukladá tri čísla typu int.

## class TransparentPanel, class TextPanel, class ImageContainer

Pomocné classes, ktoré slúžia na zobrazovanie objektov na obrazovke. TransparentPanel je potomok triedy Panel, ale umožňuje priehľadné pozadie. TextPanel je potomok triedy Panel slúžiaci špeciálne na zobrazovanie textu. ImageContainer obsahuje názov súboru, z ktorého sa má vytvoriť obrázok, jeho súradnice a veľkosť.

# Grafické prostredie

Veľká časť grafického prostredia je založená na priehľadných obrázkoch. Windows Forms však nemajú dobrú podporu priehľadnosti. Bolo teda potrebné nájsť spôsoby, ako toto obmedzenie obísť.

Základný podklad tvoria štyri panely – panel s miestnosťou a Trapulátor pre každého hráča. Ako panel pre miestnosť je použitá trieda TransparentPanel. Oproti triede Panel je v nej niekoľko zmien:

* Zmenená metóda OnPaintBackground. Zatiaľ čo trieda Panel vymaľuje pozadie bielou farbou, TransparentPanel nespraví nič.
* Zmenená metóda OnPaint. Každý TransparentPanel má List objektov ImageContainer. Pri evente OnPaint sa vykreslí každý obrázok z tohto Listu.

Pri každom pohybe nábytku alebo otvorení dverí je najskôr zmenený daný atribút v ImageContainer, a potom je použitá metóda Invalidate() iba na danú plochu, aby sa zabránilo príliš častému prekresľovaniu celej plochy, čo vedie k nestálemu obrazu.

Obrázky hráča sú objekty PictureBox. Ich poloha, veľkosť atď. sú menené priamo ako atribúty objektu. Pri pohybe hráča je ale takisto potrebné prekresliť plochu pod ním.

Trapulátor je takisto objekt typu TransparentPanel. Na ňom je odpočet času, čo je objekt typu TextPanel. TextPanel má, podobne ako TransparentPanel, niektoré zmenené vlastnosti:

* Pri evente OnPaint sa vypíše text, ktorý je uložený v atribúte text.
* Metóda UpdateText zmení hodnotu atribútu text a zavolá Invalidate(), čím sa celý panel nakreslí znova.

Trapulátor ďalej obsahuje miesto na textovú správu, čo je rovnako objekt typu TextPanel. Obrázky pascí aj pozbieraných predmetov sú uložené v objektoch ImageContainer patriacich do atribútu images daného Trapulátora.

Správy typu Loading level, Welcome a podobne sa nachádzajú v objekte PictureBox, ktorý sa zobrazuje a skrýva podľa potreby.

# Umelá inteligencia

Počítač riadi akcie svojho hráča prostredníctvom jednoduchého rozhodovacieho stromu. Na základe aktuálneho stavu pri každom ticku časovača vyberie vhodnú akciu, ktorá bude nasledovať.

Rozhodovací strom, ktorým sa riadi, je nasledovný:

* Ak je pri hráčovi iný hráč, začne s ním bojovať, pretože sa dá predpokladať, že to isté spraví protivník.
* Ak pri ňom iný hráč nie je, zistí, či má všetky predmety. Ak áno, hľadá letisko.
  + Ak v miestnosti ostávajú neprejdené dvere, zistí, či sa pri nejakých nachádza.
    - Ak áno, zistí, či sú otvorené.
      * Ak áno, prejde cez ne a uloží ich na DoorsToAirportStack.
      * Ak nie, otvorí ich.
    - Ak nie, zavolá metódu GoToLocation(súradnice prvých neprejdených dverí v poli unvisitedDoors).
  + Ak v miestnosti neostávajú neprejdené dvere, zistí, či je pri dverách na vrchu zásobníka DoorsToAirportStack.
    - Ak áno, zistí, či sú otvorené.
      * Ak áno, prejde cez ne.
      * Ak nie, otvorí ich.
    - Ak nie, zavolá metódu GoToLocation(dvere na vrchu zásobníka).
* Ak nemá všetky predmety, zistí, či je v miestnosti iný hráč.
  + Ak je, zistí, či má kufrík.
    - Ak áno, zistí, či je blízko neho.
      * Ak áno, bojuje s ním.
      * Ak nie, zavolá metódu GoToLocation(súradnice hráča).
  + Ak nie je alebo ak hráč nemá kufrík, zistí, či v miestnosti ostávajú neprehľadané objekty.
    - Ak áno, zistí, či je blízko objektu.
      * Ak áno, zdvihne ho a označí ako prejdený.
        + Ak pri ňom stratil objekt (súčasná hodnota numberOfObjects je menšia predchádzajúca), zdvihne ho znova.
      * Ak nie, ide k objektu.
    - Ak neostávajú neprehľadané objekty, prehľadáva dvere. Toto robí rovnako ako v časti hľadania dverí na letisko, rozdiel je ten, že namiesto DoorsToAirportStack používa DoorsToExploreStack.

Pokiaľ hráč má všetky predmety a nenašiel letisko, alebo prehľadal všetky objekty a nemá ich, pohybuje sa náhodným spôsobom.

# Možné rozšírenia

* Pridanie možnosti mať viacero poschodí. Základ na to je vytvorený (Position má tri súradnice, dvere vedú do miestností určených tromi súradnicami atď.). Bolo by potrebné pridať nový druh dverí (rebrík), ktorý by to umožnil.
* Zobrazovanie mapy. V originálnej verzii si hráč mohol zobraziť mapu aktuálneho poschodia, kde boli vyznačené miestnosti, v ktorých sa nachádza nejaký predmet. Zobrazením mapy stratil niekoľko sekúnd.
* Vylepšenie umelej inteligencie. V súčasnosti po vyčerpaní možností dverí a nábytku robí počítač náhodné pohyby. Toto by sa dalo zmeniť tak, že sa obnovia všetky objekty ako neprejdené. Ďalej by mohol nastavovať pasce a vyhýbať sa im. Ďalšie možné vylepšenie je vybrať najlepší možný pohyb – teda napríklad ak protihráč má všetky predmety, je zbytočné prehľadávať miestnosti. Oveľa výhodnejšie je prísť k nemu a pokúsiť sa s ním bojovať.
* Vylepšenie UI. Nájsť spôsob, ako vykresliť všetky objekty bez toho, aby vznikal na obrazovke šum.