PacMan Ultimate

Dokumentace k programu
Autor: Filip Horký

Programátorská část

ANOTACE

Jedná se o remake klasické arkády Pacman.

ZVOLENÉ ŘEŠENÍ

Po spuštění hry se hráč o citá v Menu, přesněji jeho podčásti kteoru je úvodní obrazovka. Veškerou funkcionalitu menu zajišťují procedury `OrgGame_Click()`, `selectMap_Click()`, `HighScr_Click()`, `VS_Click()` a `Settings_Click()`, starající se o vstup ve formě klinutí myši a z klávesnice. Na jeho základě nastavují proměnnou `menulayer` na hodnoty: start, menu a submenu. Aktuálně zvolená (ale dosud nevybraná) možnost v menu je uložena v proměnné `menuSelected`, jenž nabírá následujících hodnot: Settings/ VS/ Game/ SelectMap/ HighScore a k nim příslušející labely. O nastavení viditelnosti příslušných prvků se pak na základě hodnoty s níž bzla zavolána stará funkce `Menu()`. Tato funkce nejprve zneviditelní všehny viditelné prvky, jenž jsou uložené v proměnné `activeElements`. Následně zavolá příslušnou funkci jenž dostala jako parametr, která tuto proměnnou naplní novými labely, a na závěr `Menu()` zviditelní všechny nové prvky v proměnné `activeElements`.
Druhou klíčovou funkcí zajišťující funkčnost menu je fuknce `PacManUltimate_KeyDown()`. Tato funkce spravuje vstup z klávesnice a při stusknutí kláves pohybu, ESC nebo ENTER plní obdobnou funkci jako výše zmíněné funkce `..._Click()`.

Dalším podstatným aspektem programu je načtení mapy. To zajišťuje třída `LoadMap`, která je v programu volána funkcí `OrgGame_Click()`, nebo `selectMap_Click()`. Načtení mapy probíhá v několika fázích:

V první fázi funkce `LoadIt()` načte fyzický textový soubor do 2D pole znaků. Během tohoto procesu zároveň kontroluje, zda-li má načítaný soubor stanovený počet řádků a sloupců. Pokud tomu tak není, oznámí chybu formou návratu z funkce s hodnotou null. Spolu s tím program také počítá Pellets na mapě, pro usnadnění testu hratelnosti mapy v následujícím kroce. Dále také funkce `LoadIt()` zajišťuje převod čtených symbolů na pole, předdefinována ve třídě `Tiles`. Tento proces provádí funkce `TransformToTile()` a provádí ho zároveň při čtení znaků tak, že se vždy převádí symbol o řádek výše, než je symbol právě čtený. Je tomu tak právě proto, že tato funkce rozpoznává typ pole na základě všech polí jemu přilehlých. Tedy prvními 9

parametry zadanými při volání této funkce jsou kromě zpracovávaného symbolu i symboly na všech okolních polích (dosud nepřečtený symbol v pravém dolním rohu je zjištěn pomocí funkce streamReaderu `.Peek()`). Následují další dva parametry, těmi jsou již převedená pole nad a nalevo od právě zpracovávaného. Posledním parametrem je pole symbolů – slouží jako abeceda ke správnému rozšifrování textového souboru.
Nerozhodnutelná pole jsou uložené do proměnných `verticalTileStack` a `horizontalTileStack` (podle toho zda-li určují vodorovnou nebo svisou zeď), a po dokončení čtení jsou vyhodnocenz ještě jednou v opačném pořádí jak byly čteny s pomocí okolních již rozhodnutých polí.

V druhé fázi, v případě že načtení proběhlo úspěšně, funkce `IsPlayable()` otestuje reálnou hratelnost mapy. Princip testu spočívá v provedení BFS z přednastavené pozice na mapě (pacmanova startovní pozice). V tomto kroce program zjistí, zda jsou všechny Pellets na mapě z této pozice přístupné. Zároveň program testuje zda se ze zkoumaného pole dosažitelná brána do Ghost House a v kladném případě následně prohledá část pod tímto prostorem za účelem otestování jeho existence na mapě.

V případě že mapa testem projde, je vrácené výsledné 2D pole již převedených polí, spolu s počtem Pellets na mapě a pozicí Ghost House, pro pozdější použití ve hře. V opačném případě je vrácena hodnota null.

Po úspěšném načtení mapy nastává proces načítání který zajišťují dvě hlavní funkce `MakeItHappen()`, starající se o přechod mezi menu a samotnou hrou. Provádí tak nastavením několika proměnných a následným zavoláním druhé hlavní funkce `PlayGame()`.
Tato funkce se stará o načítání hry a to jak při jejím startu, tak při restartu úrovně. Mezi těmito možnostmi je rozlišováno pomocí booleanu restart, jenž je parametrem při volání této funkce. Tato funcke zajišťuje kompletní načtení mapy z 2D pole skrz funkci `RenderMap()`. Kromě toho také nastavuje počáteční stav všem herním proměnným funkcí `LoadingAndInit()`, která také zajišťuje zobrazení načítací obrazovky při procesu načítání úrovně. Tato funkce dále volá 2 procedury, `LoadHud()` a `LoadEntities()`, které jak již plyne z jejich názvu zajišťují načtení herního prostředí obsahujícího dosažené skóre a počet životů, a načtení entit jako je Pacman a duchové. Posledním krokem před prvním zavoláním herní funkce `GameLoop()` je posun Pacmana a červeného ducha o několik pixelů doleva neboť v načítací obrazovce sedí na rozmezí dvou polí, avšak při samotné hře se pohybují pouze po celých polích, je proto nutné je posunou právě do středu jednoho z oněch polí.

Během samotné hry je prvkem `PacUpdater` každých 100 ms volána herní funkce `GameLoop()` s příznakem `isPacman` = true, zajišťující kompletní update entity pacman. Obdobně pak prvek `GhostUpdater` zajišťuje update duchů a to též periodicky každých x sekund, kde x je přímo závislé na intervalu `PacUpdateru` a současné úrovně. Tímto způsobem je ve hře suplována funkce herního cyklu tak, aby nedošlo ke ztrátě kontroly nad oknem.

Funkce GameLoop() se stará o kompletní Update mapy entit na ní. To zajišťuje voláním funkce Update(), která se pak dále dělí na funkce UpdateMove(), UpdateEatEmTimer() a UpdateEatPellet().
První z nich, UpdateMove(), nastavuje entitám novou pozici na zákadě uživatelského vstupu zpracovávaného již zmíněnou funkcí PacManUltimate_KeyDown(), nebo příslušným AI algoritmem přednastaveným při inicializaci entit a nacházejícím se ve třídě DefaultAI. Zbylé dvě funkce UpdateEatEmTimer() a UpdateEatPellet() poskytují časovač pro Pacmanův stav excitace a kontrolují zda-li nedošlo k sebrání Pellet a v kladném případě vykonají příslušnou aktualizaci skóre, případně spustí časovač stavu Pacmanovy excitace, čímž při příští iteraci herní funkce zároveň dojde i k jejímu kompletnímu aktivování.
Další podstatnou funkcí funkce Update() je postupné vypouštění duchů z Ghost House v závislosti na příslušném časovači, jehož hodnota se snižuje s každým cyklem, nejedná se tedy o časovač v pravém slova smyslu, nýbrž spíše o čítač iterací. Veškerá zmíněná funkcionalita je navíc rozdělena do dvou neuplně disjunktních oddílů. První zajišťuje Update entitz pacman, zatímco druhý entit duchů. Vykonávaný oddíl je vybrán dle proměnné isPacman předané funkcí GameLoop().

Druhým úkolem funkce GameLoop() je zpracování výstupu funkce PacManUltimate_KeyDown() pro jeho snadné pozdější použití při aktualizaci hráčovi pozice. V principu jde o to, že se každý validní vstup udržuje pouze po určitý stanovený počet iterací, což zajišťují proměnné NewDirection1 a NewDirection1 obsahující vstup hráčů, keyPressed1 a keyPressed2, indikující zda-li došlo ke stisknutí validní klávesy a keyCountdown1 a keyCountdown2, poskytující právě ony „časovače“ pro, případně oba, hráče. Udržování těchto čítačů iterací zajišťuje procedura KeyCountAndDir().

Další část funkce GameLoop() zajišťuje kontrolu počtu sesbíraných Pellet a případně spouští další úroveň / ukončuje hru.

Poslední část funkce GameLoop() spouští příslušný časovač (PacSmoothTimer/ GhostSmoothTimer), starající se animaci přechodu entit mezi poli skrz periodické volání metod GhostSmoothTimer_Tick() a PacSmoothTimer_Tick().

K ukončení hry slouží procedura EndGame(), starající se o kompletní smazání veškerých textur, případné uložení dosaženého skóre skrz funkce třídy HighScoreClass a opětovné načtění herního menu.

POUŽITÉ DATOVÉ STRUKTURY

Nejpoužívanější datovou strukturou napříč programem je enum a vyskytuje se jich zde hned několik:

- enum Direction.nType { LEFT, RIGHT, UP, DOWN, DIRECTION}
  Slouží k uložení směru pohybu jednotlivých entit po herním poli.

- `enum DefaultAI.nType { PLAYER1, PLAYER2, HOSTILEATTACK, HOSTILERETREAT, CANBEEATEN, EATEN }`
  Slouží ke snadnému rozeznání entit z pohledu programu a určuje druh AI algoritmu pro pohyb entity po poli.
- `enum Tile.nType { DOT, POWERDOT, FREE, GATE, LWALLDOUBLE, RWALLDOUBLE, TWALLDOUBLE, BWALLDOUBLE, LWALLSINGLE, RWALLSINGLE, TWALLSINGLE, BWALLSINGLE, TLCURVEDOUBLE, TRCURVEDOUBLE, BRCURVEDOUBLE, BLCURVEDOUBLE, TLCURVESINGLE, TRCURVESINGLE, BRCURVESINGLE, BLCURVESINGLE, HTBDTILE, VTBDTILE, TILE };`
  Slouží k rozeznání jednotlivých herních polí a snadnou manipulaci s nimi.

Dalšími pužitými datovými strukturami jsou:

- `Tuple<Tile.nType?[][], int, Tuple<int, int>> Map;`
- `Tuple<Tile[][], int, Tuple<int, int>, Color, List<Point>> Map;`
  N-tice obsahující informace o načtené mapě. Prvním prvkem je 2D pole polí, druhým prvkem je počet Pellets na mapě, třetím prvkem je dvojice souřadnic Ghost House resp. start pozice červeného ducha, čtvrtým parametrem je barva mapy, načtená ze seouboru a posledním parametrem je seznam pozic PowerPellets pro umožnění jejich blikání ve hře.
  Veškeré informace jsou získány skrz třídu `LoadMap` ve funkci `MakeItHappen()` z načteného souboru.
- `Tuple<int, int, PictureBox, Direction.nType, DefaultAI>[] Entities;`
  N-tice obsahující informace o entitách na herním poli. První dva prvky jsou X a Y pozice entity na mapě měřená v polích. Toto řešení bylo na místo nabízeného `Tuple<int, int>` zvoleno kvůli četnosti přístupu k jednotlivým souřadnicím entity, čímž došlo ke zkrácení zápisu z případného Entity.item1.item1/2 na Entity.item1/2.
  Třetím prvkem struktury je `PictureBox`, jedná se o obrázek symbolizující danou entitu na herním poli. Dalším prvkem je enum `Direction.nType` jejhož význam je již uveden jednou výše, následovaný instancí třídy `DefaultAI` sloužící jako identifikátor chování dané entity pro program.

Mezi dalšími složitějšími proměnnými lze jmenovat:

- `WMPLib.WindowsMediaPlayer[] SoundPlayers;`
  Instance Windows Media Player sloužící k umožnění spouštění překrývajících se zvuků – například při požírání Pellets Pacmanem lze dosáhnout požadované rychlosti přehrávání zvuku pouze použitím více střídajících se přehrávačů zároveň.
- `System.Media.SoundPlayer MusicPlayer`
  Defaultní přehrávač zvuků dostupný ve WinForms ideální pro přehrávání doprovodné hudby, zejména díky své vestavěné funkci PlayLooping(), umožňující přehrávat zvuk ve smyčce dokud není externě narušen.
- `Tile[][] MapFresh;`
  2D pole políček uchovávající informaci o současném stavu herního pole.
- `DefaultAI[] DefaultAIs;`
  Pole s pevně nastavenými druhy AI pro jednotlivé nepřátelské entity.
  Slouží pro obnovení původního `DefaultAI.nType` například po vypršení Pacmanovy excitace.

POUŽITÉ FUNKCE

Dokumentace obsahuje pouze účel a hrubé nastínění podstaty funkcí. Pro podrobnější popis funkcí nahlédněte do poznámek v kódu u jednotlivých funkcí.

Funkce použité pro počáteční inicializaci formuláře:

- `public PacManUltimate()`
  Funkce provádějící primární inicializaci formuláře při startu programu.
- `private void PacManUltimate_Load(object sender, EventArgs e)`
  Funkce provádějící primární načtení všech ovládácích prvků formuláře.
- `private void PacManUltimate_KeyDown(object sender, KeyEventArgs e)`
  Funkce zpracovávající hráčův vstup. Obsahuje rozdělené části pro vstup z menu a vstup ze hry.
- `private void CheckMinimization(object sender, EventArgs e)`
  Funkce kontrolující zda-li nedošlo k minimalizaci okna.

Funkce použité pro zajištění funkčnosti menu:

- `private new void Menu(Action Menu_Func)`Funkce simulující menu jednoduchým Zviditelňování a skrývání příslušných ovládacích prvků na základě větve menu ve které se hráč právě nachází.
- `private void OrgGame_Click(object sender, EventArgs e)`
  Načte přednastavenou originální mapu a spustí hru skrze MakeItHappen funkci.
- `private void AdvancedLdBut_Click(object sender, EventArgs e)`
  Zviditelní prvky, čímž umožní zadání vlastní abecedy nahrávané mapy.
- `private void SelectMap_Click(object sender, EventArgs e)`
  Otevře dialogové okno a pokusí se načíst v něm vybranou mapu. V případě úspěchu dále postupuje obdobně jako funkce OrgGame_Click.
- `private void MusicButton_Click(object sender, EventArgs e)`
  Změní hodnotu příslušné boolean proměnné a změní barvu prvku na základě hodnot onoho booleanu.
- `private void SoundsButton_Click(object sender, EventArgs e)`
- `private void VS_Click(object sender, EventArgs e)`
  Nastaví hodnotu booleanu Player2 a zavolá selectMap_Click.
- `private void Settings_Click(object sender, EventArgs e)`
  Nastaví aktivní menu větev na Settings a zavolá menu.
- `private void HighScr_Click(object sender, EventArgs e)`
  Nastaví aktivní menu větev na HighScore a zavolá menu.
- `private string CharListToString(List<char> source)`
  Funkce pro naplnění řetězce zadanými symboly oddělenými pomocí ‘;’.
- `private void MoveInMenu(int delta)`
  Zajistí zvýraznění příslušné nabídky menu při pohybu skrz vstup z klávesnice.
- `private void MoveInSettings()`
  Obdoba MoveInMenu jen pro Settings.
- `private void SoundsButton_MouseEnter(object sender, EventArgs e)`
  Zvírazní SoundsButton v Settins při příchodu myši.
- `private void MusicButton_MouseEnter(object sender, EventArgs e)`
  Zvírazní MusicButton v Settins při příchodu myši.
- `private void HighlightSelected(Label prevLabel, Label newLabel)`
  Funkce měnící zvírazvení vybrané možnosti v menu.

- `private void Hover(object sender, EventArgs e)`
  Funkce zajišťující změnu barvy příslušného textu v menu po najetí myši.
- `private void MenuKeyDownHandler (KeyEventArgs e)`
  Menu větev obsluhy vstupu z klávesnice.

Funkce použité pro přechod mezi menu a hrou:

- `private void RenderMap(Tile[][] tiles, Color color)`
  Funkce zajišťující načtení, nastavení a umístění všech herních polí a příslušných obrázků na mapu.
- `private void DeepCopy(Tile[][] source, ref Tile[][] destination)`
  Vytvoří kopii zdrojového 2D pole bez referencí, aby bylo možmé ukládat aktuální stav herního pole, ale zároveň při novém načtení obnovit původní vzhled.
- `private void placePictureBox(PictureBox pic, Image image, Point point, Size size)`
  Fyzicky umístí zadaný obrázek do mapy.
- `private void placeLabel(Label label, string text, Color color, Point point, Font font)`
  Fyzicky umístí label do mapy.
- `private void LoadEntities()`
  Funkce která připraví a nahraje všechny entity. Dále nastaví jejich jména pro snadnou budoucí programovou manipulaci a všem přednastaví počáteční směr.
- `private void LoadHud(int hp)`
  Funkce která nahraje skóre, HighScore a životy. Životy jsou uloženy v poly a umístěny na pevné pozice. V závislosti na počtu životů jsou dále pouze zviditelňovaná příslušná pole.
- `private void LoadingAndInit(Label loading, Label levelLabel, Color color)`
  Procedura sloužící k jednoduchému přednastavení proměnných a zobrazení načítací obrazovky. Pro více informací o jednotlivých proměnných nahlédnětě do poznámek v kódu u této funkce.
- `private Color ChooseRandomColor()`
  Funkce sloužící pro výběr náhodné barvy v případě jejího nespecifikování v načítaném souboru.
- `private void PlayAnimation()`
  Funkce sloužící k přehrávání krátkých semi-náhodných animací mezi úrovněmi.
- `Private async void PlayGame(bool restart)`
  Zajišťuje prvotní vykreslení mapy a obecné připravení programu na začátek úrovně. Má na starost READY label na začátku každého pokusu. Opravuje pozice aktivních entit kvůli jejich prvotnímu umístění na rozmezí mezi pole.
- `private void MakeItHappen(LoadMap loadMap)`
  Most mezi menu a hrou. Vypíná funkce menu a přiřazuje vstup herní větvi PacManUltimate_KeyDown pomocí booleanu gameOn. Dále inicializuje mapu a volá funkci pro start úrovně.

Funkce použité pro samotný běh hry:

- `private void GameKeyDownHandler(KeyEventArgs e)`
  Herní větev obsluhy vstupu z klávesnice.
- `private void EndGame()`
  Ukončuje hru zastavením updateru, zapnutí funkcí menu, ukládá HighScore.
- `private async void KillPacman()`
  Vyřizuje událost vzniklou zabitím Pacmana nějakou ze zbývajících entit.
- `private void UpdateHud(int? score, Label box)`
  Aktualizuje vybraný prvek skóre.
- `private bool IsDirectionFree(int y, int x, Tuple<int, int, PictureBox, Direction.nType, DefaultAI > entity)`
  Kontroluje je-li směr kam entita míří volný.
- `private void SetToMove(ref Direction.nType newDirection, ref Tuple<int, int, PictureBox, Direction.nType, DefaultAI> entity)`
  Kontroluje zda-li je směr kam by se entita chtěla posunout volný a pokud ano, tak nastaví její směr a vynuluje příslušnou proměnnou.
- `private void CanMove(ref Tuple<int, int, PictureBox, Direction.nType, DefaultAI> entity)`
  Kontroluje zda-li entita může pokračovat dále směrem kterým míří, jinak ji zastaví.
- `private void UpdatePicture(bool change, int dx, int dy, ref Tuple<int, int, PictureBox, Direction.nType, DefaultAI> entity)`
  Přiřadí entitě správnou texturu v závislosti na jejím směru a stavu (těmi jsou vyjma normálního také stav EATEN a CANBEEATEN).
- `private void CorrectPicture(ref Tuple<int, int, PictureBox, Direction.nType, DefaultAI> entity)`
  Umístí obrázek na místo odpovídající jeho pozici v TileMap.
- `private void MoveEntity(bool change, int entX, int entY, int invX, int invY, ref Tuple<int, int, PictureBox, Direction.nType, DefaultAI> entity)`
  Nastaví novou pozici entity v polích i fyzickou polohu obrázku.
- `private void MoveIt(ref Tuple<int, int, PictureBox, Direction.nType, DefaultAI> entity)`
  Nejsvrchnější vrstva procesu pohybu entity. Zajišťuje, aby se i v případě teleportace, která se z pohledu programu jeví jako pohyb jiným směrem než pacman reálně směřuje, umístila správná textura.
- `private bool IsAtCrossroad(int x, int y)`
  Kontroluje zda-li entita dosáhla křižovatky (myšleno cesty s více jak 2 odbočkami). Odchytává a ignoruje výjimku v případě že se pacman chystá teleportovat a funkce se dožaduje přístupu k poli mimo velikost mapy.
- `private void UpdateMove(bool isPacman)`
  Zajišťuje kompletní pohyb všech entit a volá funkce pro pohyb jednotlivých entit. Také kontroluje zda-li nedošlo ke kontaktu Pacmana s nějakou z ostatních entit a to pomocí funkce CheckCollision.
- `private void PlayWithSoundPlayer(string file)`
  Přehraje zvuk specifikovaný parametrem soudplayerem jenž je právě na řadě.
- `private void UpdateEatEmTimer()`
  Poskytuje časovač pro Pacmanův stav excitace a zajišťuje přepnutí všech duchů zpět do původního stavu po jejím skončení.
- `private void UpdateEatPellet()`
  Kontroluje zda-li Pacman nedosáhl pole s Pellet nebo Poewr Pellet. V kladném případě zajistí vyprázdnění tohoto pole, zvýšení skóre a v případě, že se jednalo o Power Pellet také spuštění Pacmanovi excitace nastavením EatEmTimer a přivedením duchů zpět do normálu, aby mohli být znovu pozřeni.
- `private void SetGhostFree(int ghostNum)`
  Funkce jenž je zodpovědná za vypouštění duchů z Ghost House a nastavení jejich počáteční pozice a směru.

- `private void CheckCollision()`
  Kontroluje zda-li nedošlo ke kolizi mezi pacmanem a nějakou z ostatních entit.
- `private void UpdateGame(bool isPacman)`
  Funkce zodpovědná za kompletní aktualizaci herního pole. Je volána asznchronně pacmanem i duchy. Postupně volá funkce UpdateMove, UpdateEatEmTimer a UpdateEatPellet. Dále zajišťuje udělení bonusového života při dosažení nastaveného skóre. Také obsluhuje časovač na postupné vypouštění duchů.
- `private void Blink()`
  Zajišťuje periodicke blikání herních prvků.
- `private void RedrawPellets()`
  Stará se o opětovné vzkreslení Pellets odstraněných pohybem duchů.
- `private void KeyCountAndDir(ref Direction.nType direction, ref int keyCountdown)`
  Kontroluje zda-li směr v němž se hráč chce posunout je volný a stará se o časovač, zajišťující pamatování si hráčovi volby směru po daný časový úsek.
- `private void GameLoop(bool pacman)`
  Nejsvrchnější vrstva instrukcí prováděných každý Frame. Poskytuje časovače pro vstup obou hráčů, volá Update funkci a kontroluje zdali hráč již nesesbíral všechny Pellets na herním poli. V takovém případě v závislosti na hrané úrovni a herním módu, spustí úroveň následující, či ukončí hru.
- `private async void EndLevel()`
  Zajišťuje zamrznutí hry po dokončení úrovně a rozblikání mapy.
- `private void PacUpdater_Tick(object sender, EventArgs e)`
  Vyvolává kompletní update pacmana.
- `private void GhostUpdater_Tick(object sender, EventArgs e)`
  Vyvoláva kompletní update duchů.
- `private void PacSmoothTimer_Tick(object sender, EventArgs e)`
  Provádí animaci hladkého pohybu pacmana po herním poli.
- `private void GhostSmoothTimer_Tick(object sender, EventArgs e)`
  Provádí animaci hladkého pohybu duchů po herním poli.
- `private Point GetDeltas(byte index)`
  Vrací delty pro animaci pohybu obrázků entit.

Funkce použité pro rozpoznání mapy:

- `public LoadMap(.......)`
  V závisloti na parametrech s nimiž je zavolána spustí nahrávací proces s již přednastavenou abecedou symbolů, nebo abecedou specifikovanou v parametrech.
- `private Tuple<Tile[][], int, Tuple<int, int>,System.Drawing.Color, List<System.Drawing.Point>> LdMap(string path, char[] symbols)`
  Vrchní vrstva procesu nahrávání mapy. V případě úspěšného nahrání předá mapu testovací funkci IsPlayable, která otestuje její hratelnost.
- `private Tuple<char[][],Tile[][],int, System.Drawing.Color, List<System.Drawing.Point>> LoadIt(string path,char[] symbols)`
  Funkce jenž zajišťuje nahrání mapy z textového souboru a zároveň vytváří její alternativu s poli jako `Tile[][]`. Přesný postup algoritmu je vysvětlen v kódu.
- `private bool PerformReverseEvaluation(ref Tile[][] tileMap, Stack<System.Drawing.Point> hStack, Stack<System.Drawing.Point> vStack)`
  Vyhodnotí dosud nerozhodnuté pole s pomocí okolních již rozhodnutých polí.
- `private System.Drawing.Color ParseColor(string line)`
  Načte barvu z prvního řádku vstupního souboru.

- `private Tuple<bool, Tuple<int, int>> IsPlayable(char[][] map, char[] symbols, int numOfDots)`
  Otestuje zda-li je nahraná mapa hratelná, tedy: obsahuje Ghost House, startovní pozice Pacmana je volná a všechny Pellets jsou z této pozice dosažitelné, spolu s Ghost House. Provádí tak za pomocí klasického BFS.
- `private Tile TransformToTile( char tile, char tileLeft, char tileRight, char tileUp, char tileDown, char BLCorner, char TLCorner,char TRCorner, char BRCorner, Tile upperTile, Tile leftTile, char[] symbols)`
  Tato na volací parametry poměrně náročná funkce zodpovídá za přeložení symbolů z načítané mapy do `Tile`. Činí tak za pomoci rozhodovacích algoritmů pracujícím na základě porovnávání polí přilehlých zkoumanému poli.

Ostatní funkce programu:

- class Direction
  - `public Tuple<int,int> DirectionToTuple(nType direction)`
    Funkce pro převod z `Direction.nType` na dvojici přírůstků v osách x a y.
  - `public nType TupleToDirection(Tuple<int,int> tuple)`
    Funkce reverzní k funkci DirectionToTuple.

- class Tile
  - `public nType ReturnType(string tile)`
    Funkce pro převod vybraných klíčových slov na `Tile`.
  - `public Tile(nType tileType)`
    Vytvoří instanci `Tile` stejného typu jako vstupní hodnota.
  - `public void DrawTile(Graphics g, Point location, Color color)`
    Vykreslí políčko na dané pozici, danou barvou.
  - `public void FreeTile(Graphics g, Point location, Color bkgColor)`
    Uvolní políčko nakreslením čtverce barvou pozadí na dané místo.
  - Třída také obsahuje definice typů polí jako křivek

- class HighScoreClass
  - `public int LoadHighScore()`
    Načte HighScore z ImportantLine řádku souboru config.bin a převede ho z binárního do desítkového zápisu.
  - `public int SaveHighScore(int newHighScore)`
    Ukládá dosažené HighScore do souboru config.bin. Dosažené HighScore převede do binárního zápisu a uloží ho na ImportantLine řádek souboru. Na řádky před a za jsou vygenerována náhodná binární data (pro zmatení nepřítele).

- class DefaultAI
  - `public DefaultAI(nType state, int tsc, int tsr)`
    Vytvoří instanci DefaultAI s AI nastaveným na state. Pro pozdější použití vyžaduje i zadání šířky a výšky mapy v polích.
  - `public Direction.nType NextStep(Tuple<int,int> position, Tuple<int,int> target, Direction.nType direction, Tile[][] map)`
    Volá příslušnou funkci pro vypočítání nového směru entity na základě typu AI. Součástí volací konvence funkce je také zadání cílového políčka pro případnou snadnou pozdější implementaci složitějších AI algoritmů.
  - `public Direction.nType RandomAI(Tuple<int, int> position, Direction.nType direction, Tile[][] map)`
    Základní AI vracející nový směr na základě náhody. Zjistí jaké ze směrů jsou volné a z nich na základě náhody jeden vybere. Směr ze kterého entita přišla vybere pouze v případě, kdy žádný jiný není volný.
  - `private bool CanAdd(Tile tile)`
    Testuje zda-li je zkoumané pole volné.

- Kromě zmíněných funkcí třída obsahuje i funkce pro chování AI v různých stavech. Tyto funkce mají tvar:
- `virtual public Direction.nType [Název Funkce](Tuple<int, int> position, Tuple<int, int> target, Direction.nType direction, Tile[][] map)`

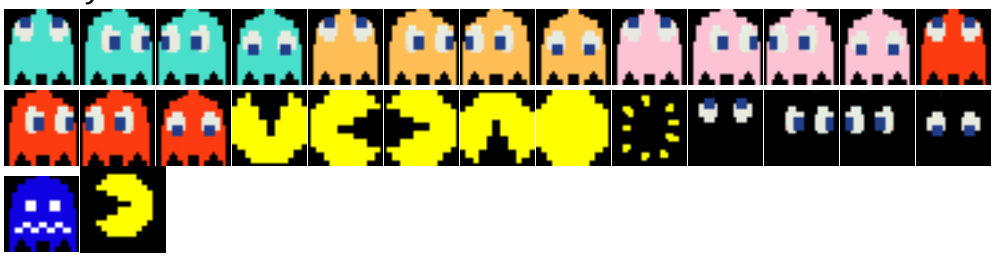
CO NEBYLO DODĚLÁNO

Složitější AI – do volání funkce pro AI bylo zahrnuté i cílové pole `Tuple<int,int> target` pro případné budoucí usnadnění tvorby složitějšího algoritmu AI, založeného na bázi hledání nejkratší cesty. Základním náhodným algoritmem pro AI je tato proměnná ignorována.

Zadávání vlastní symboliky použité v načítání mapy nefunguje podle intuice uživatele kvůli inkonzistenci mezi ASCII tabulkou a dvojicí KeyCode a KeyValue ve WinForm – chybí funkce pro překlad mezi použitými „abecedami“.

POTŘEBNÉ SOUBORY

Program počítá s následujícím:

- Podsložka obsahující sloužku se spouštěným .exe souborem obsahuje také složku textures a sounds.
- Déle je v této složce soubor OriginalMap.txt s mapou, která bude automaticky načítána při spuštění „Original Game“.
- Program také používá soubor config.bin uložený v téže složce pro ukládání a načítání dosaženého nejvyššího skóre. Pro svoji funkčnost ho však, na rozdíl od předcházejích, nevyžaduje.
- Obsah složky textures:
  - 
  - Obrázky jsou ve formátu 28 x 28, s výjimkou posledního, ten je ve formátu 32 x 32.

- Obsah složky sounds:
  - pacman_beginning.wav
  - pacman_death.wav
  - pacman_eatensiren.wav
  - pacman_eatghost.wav
  - pacman_extrapac.wav
  - pacman_chomp.wav
  - pacman_intermission.wav
  - pacman_powersiren.wav
  - pacman_siren

Uživatelská část

MENU

Hra se otevře v automaticky nastavené velikosti okna, která je pevná. Uživatel se rázem octne na úvodní obrazovce odkud je po stisknutí libovolné klávesy (kromě ESC) veden do menu.

V Menu se hráč pohybuje pomocí myši nebo W/S nebo šipek. Po zvolení příslušného ovládacího prvku daný prvek změní barvu, aby upozornil uživatele, že je zvolen. Výběr daného ovládacího prvku uživatel provede stisknutím levého tlačítko myši nebo klávesy ENTER.

Pro návrat z větví menu které tuto funkci umožňují, jako například HighestScore nebo Settins, zpět do hlavního menu, nebo pro návrat z hlavního menu zpět na úvodní obrazovku, stikněte klávesu ESCAPE.

V případě chyby při načítání mapy je uživatel veden do Select Map větve menu. V případě snahy o načtení mapy pužívající jinou „abecedu“ než je ta uvedena v podrobnostech zobrazených v tomto submenu, klikněte myší na Advanced Load a podle instrukcí zadejte danou abecedu v uvedeném pořadí. Pro opravení symbolu při chybném zadání stiskněte klávesu BACKSPACE a zadejte nový symbol.

HRA

Hráč ovládá Pacmana – žlutý kruh s výsečí pomocí kláves WASD nebo šipek klávesnice. Smyslem hry je sesbírat všechny puntíky, dále Pellets, dříve než je hráč zabit nepřátelskými duchy.

Po sesbírání všech Pellets je mapa vždy překreslena a načtena znovu s postupně se zvyšující úrovní obtížnosti.

Podstatou hry je vydržet co nejdéle, tedy dosáhnout pokud možno co nejvyššího skore.

Na herním poly se kromě normálních Pellets může objevit i jejich větší varianta, takzvaná Power Pellet. Ta hráči po sebrání garantuje dočasnou schopnost pojídat duchy, čímž hráč získá jednak krátký odpočinek, druhak nemalé odměny v podobně skóre za každého pozřeného ducha.

Kromě základní hry je k dispozici i varianta hry pro dva hráče, takzvaný VS mód. V této alternaci jeden z hráčů ovládá červeného ducha a jeho úkolem je spolu s ostatními třemi duchy, ovládanými stále pomocí AI, zabránit pacmanovi v sesbírání všech Pellets. V této variantě hry je pacman ovládaný klávesami WASD a červený duch pomocí šipek klávesnice.