P3Kmen

A program használata:

A program elindításakor a konzol jelenik meg, ekkor a felhasználó kiválasztja a felbontást, majd azt, hogy teljes képernyőn szeretné-e futatni a játékot. A főmenü 4 menüvel rendelkezik: egyjátékos, többjátékos, dicsőség lista, kilépés. A felhasználó(k) a nyíl billentyűk segítségével választhatnak a menüpontok között és az enter lenyomosával véglegesíthetik azt.

Egyjátékos:

A menü kiválasztása után a játékos kiválaszthatja, a nehézségi szintet, ami meghatározza, hogy mennyi élettel kezd. (könnyű = 5, normál = 3, nehéz = 1).

A játékos a nyíl billentyűk segítségével irányíthatja karakterét, a falakkal teli pályán.

A játékmód célja a legtöbb pont szerzése. A pontokat a játékos több módon szerezhet. A pályán található pontok felvételével, vagy a szörnyek megevésével.

A pályán találhatóak pontok és módosítók. A módosítók felvételével a játékos ideiglenesen megeheti a szörnyeket pontokért cserébe.

A játékost különböző szörnyek üldözik a pályán. Ha valamelyik szörny elkapja a játékost, akkor a játékos a pálya egy véletlenszerű pontján éled újra és elveszít egy életet.

A játék véget ér, ha elfogytak a játékos életei vagy megszerezte a pályán fellelhető összes pontot.

A játék végeztével a program kiírja a játékos pontszámát a képernyő aljára, fölé pedig az eddigi 10 legjobb elért eredményt.

A játékos eredmény: szerzett pontok/kezdő életek száma.

Ha a játékos pontszáma elegendő a dicsőséglistára való felkerülésre, akkor a program bekéri a felhasználó által megadott nevet, majd a pontszáma felkerül a megfelelő helyre a dicsőség listán.

A dicsőséglista után a program visszamegy a főmenübe.

Kétjátékos:

A játék menete megegyezik a program egyjátékos módjával. Különbségek:

A nehézségi szintválasztás helyett a játékosok kezdő életpontjait és a számukat kell kiválasztani.

1. játékos irányítása: nyíl billentyűk segítségével.

2. játékos irányítása: ’w’ ’a’ ’s’ és ’d’ billentyűk segítségével.

A játék véget ér, ha mindegyik játékosnak elfogy az életpontja, vagy minden pályán lévő pont elfogyott.

Ezek után program kiírja a játékos eredményét csökkenő sorrendben, majd a program visszalép a főmenübe.

Dicsőséglista:

A program kiírja az eddig elért 10 legjobb eredményt.

Kilépés:

A program kilép.

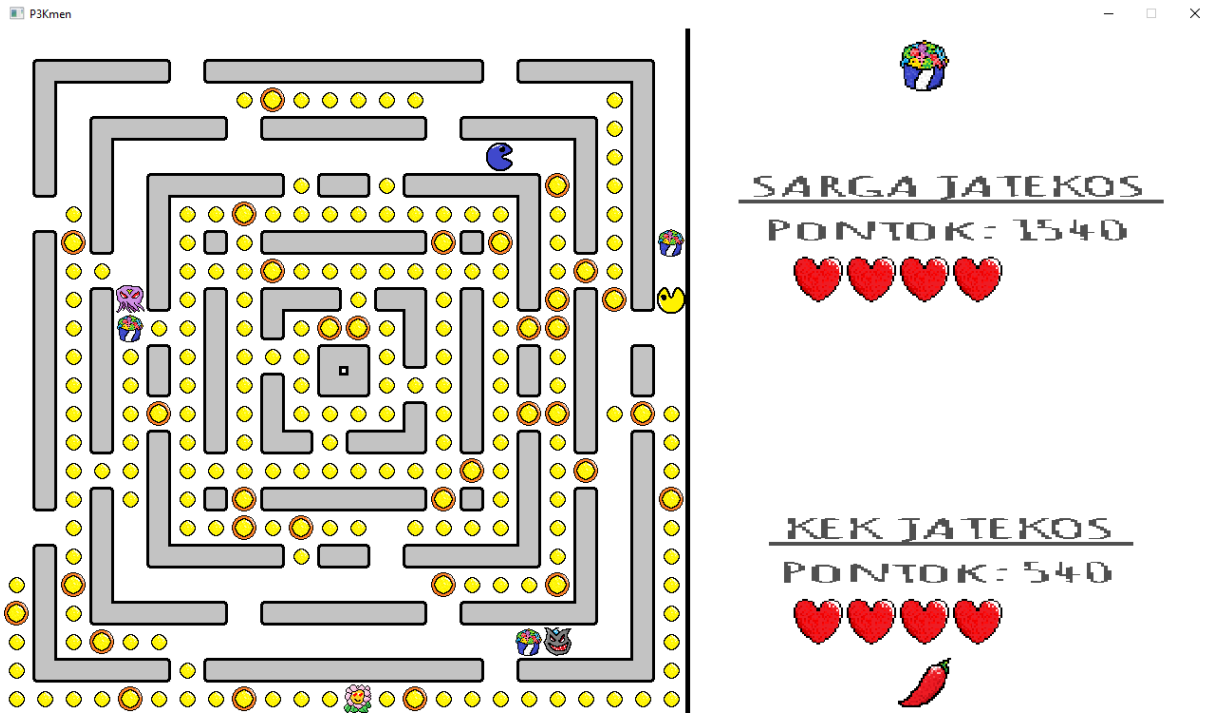
Ki- és bemenet:

A program a P3Kmen.exe fájl mappájában a ’levels’ nevű mappában levő level_’palyánév’.txt alakú fájlokból tölti be a pályákat. Az ’eredmények.txt’ az indító fájlal egy mappában található, és az eredmények onnan kerülnek betöltésre.

Sípos Áron Dávid
Neptun: EKAUKB

2018. december 07.

Képek a programból:



Sipos Áron Dávid
Neptun: EKAUKB

2018. december 07.

Programozói dokumentáció:

Szükséges könyvtárak: SDL2 SDL_image, SDL_ttf, SDL_mixer

Szükséges fájlok: a sources mappa tartalma a textúrákhoz, hangokhoz, betűtípushoz.

Licenszfájlok megtalálhatók a sources mappában a megfelelő fájlok mellett.

A képek saját készítésűek.

sources mappa:

- fonts

Amatic-Bold.ttf

- sounds:

point_big_eaten.wav

nom_nom.wav

monster_eaten.wav

menu_click.wav

menu_choose.wav

Loyalty_Freak_Music_-_Sweat_Time.wav

freeze.wav

chili.wav

point_normal_eaten.wav

- textures:

Adatszerkezetek:

A program legfontosabb structjai:

- typedef struct renderAssets

  Ez magába foglal mindent, ami rendereléshez, hangeffektekhez kell. Ez nagyon előnyős mert így nagyon könnyen át lehet adni mindezeket a függvények között.

- typedef struct Player

  Egy játékos adatait foglalja magába így könnyen nyomon követhető minden

- typedef struct Monsters

  A három szörny adatait fogja össze, így nagyon könnyű a kezelésük.

- typedef struct Level

  Talán a legfontosabb, a pályát és adatait fogja össze. Magát a pályát egy TileType 2D tömb adja meg, emellett ez a struktúra tárolja az időzítőket a játék eseményeihez.

Struktúrák és enumok:

- typedef struct Coordinate

  Struktúra egy koordináta eltárolásához

- typedef enum Resolution

  Enum a felbontáshoz

- typedef struct windowData

  Struktúra amely az program ablak adatait tárolja

- typedef struct initRenderer

  Struktúra ami egybefogja a renderert

- typedef struct initSounds

  Struktúra ami egybefogja a hangokat

- typedef struct initTexture

  Struktúra ami egyfogja az összes textúrát

- typedef struct initFont

  Struktúra ami eltárolja egy SDL_ttf betűtípus adatait

- typedef struct renderAssets

  Struktúra ami eltárolja a megjelenítéshez és a hangeffektek adatait

- typedef struct LevelDimensions

  Struktúra a pálya dimenzóinak

- typedef struct initMainMenuCoords

  Összefogja a főmenü rendereléséhez szükséges SDL_Rect változókat

- typedef struct initTopListCoords

  Összefogja a dicsőséglista rendereléséhez szükséges SDL_Rect változókat

- typedef struct initGetDifficultyCoords

  Összefogja a nehézség választó menü rendereléséhez szükséges SDL_Rect változókat

- typedef enum TileType

  Enumerizáció a pálya lehetséges elemihez

- typedef enum Direction

  Enumerizáció az irányoknak

- typedef struct MoveEnable

  Struktúra amely tárol : egy irányt, koordinátát, és engedélyezést és hogy játékos-e a lépés

- typedef enum MonsterState

  Enumerizáció a szörny állapotához

- typedef enum PlayerState

  Enumerizáció a játékos állapotához

- typedef struct Player

  Struktúra mely magába foglalja egy játékos adatait

- typedef struct Monster

  Struktúra mely magába foglalja egy szörny adatait

- typedef struct Monsters

  Struktúra amely magába foglalja 3 szörny adatait

- typedef enum Difficulty

  Enumerizáció a nehézségi szinthez

- typedef struct eventTimeAndCondition

  Struktúra amely egy event idejét és állapotát

- typedef struct TimeHandler

  Struktóúra amely tárolja a játékban lévő eventeket

- typedef struct Level

  Struktúra, mely magába foglal egy pályát/*

- typedef enum MainMenuSelection

  Enumerizáció amely tárolja a főmenu lehetséges menüjeit/*

- typedef struct Place

  Enumerizáció a toplistához

Függvények

main.c :

- void singlePlayer(renderAssets *renderAsset)

  Egyjátékos mód

- void multiPlayer(renderAssets *renderAsset)

  Többjátékos mód

- void topList(renderAssets *renderAsset)

  Dicsőség lista kiírása

- void finalScoreToplist(renderAssets *renderAsset, int score)

  Dicsőség lista és adott pont kiírása, ha a pont elegendő a listára felkerüléshez a program bekéri a játékos nevét és felkerül a toplistára, az eredmények.txt-ben is

render.c :

- initRenderer initRender(windowData windowResolution)

  Elindítja az SDL-t, alap felbontás: 1280x720, windowData struktúra alapján beállítja a felbontást és a teljes képernyős módót

- initTexture loadTexture(initRenderer *renderer)

  Az *initTexture* struktúrát feltölti textúrákkal a static SDL_Texture *loadImage függvény segítségével

- initFont loadFont(const char *filePath, int fontSize)

  Megadott elérési útvolnallal betölt egy ttf fájlt fontSize méretben

- initSounds loadSounds(void)

  Betölti a hangokat, ha nem létezik a fájl akkor a hangra mutató pointer NULL

- SDL_Texture *loadImage(const char *picture, initRenderer *renderer)

  .bmp kiterjesztésű fájlokból textúrákat csinál

- void sdlClose(renderAssets *renderAsset)

  Bezárja az SDL-t

- void drawMainMenu(renderAssets *renderAsset, MainMenuSelection selection, initMainMenuCoords mainMenuCoord)

  Kirajzolja a főmenüt

- void renderTopList(renderAssets *renderAsset, initTopListCoords topListCoords, Place **topList, int placementNumber)

  Kirajzolja a toplistát

- void renderText(renderAssets *renderAsset, SDL_Rect place, const char* string)

  string stringet kirajzolja a place álltal megadott helyen. méretben

- void placeSelectionRectForRender(renderAssets *renderAsset, SDL_Rect placement, int pixelHeight)

  Kirajzolja a kiválasztást jelző csíkot, de nem jeleníti meg, a csík magassága pixelHeight,

- void renderScoreInputScreen(renderAssets *renderAsset, const char* nameString)

  Kirajzolja a név bekérés képernyőt, ahol a nameString a nevet tartalmazza

- void renderMapWithoutEntities(Level *level, renderAssets *renderAsset)

  Kirajzolja a level-ben megadott pályát a szörnyek és játékosok nélkül

- void renderMapJustEntities(Level *level, renderAssets *renderAsset, Player *playerOne, Player *playerTwo, Monsters *monsters)

  Kirajzolja a level-ben megadott pályán szörnyeket és játékosokat

- Difficulty getDifficulty(renderAssets *renderAsset)

  Kirajzolja a menüt a nehézség választáshoz, majd visszatér a kiválasztott nehézséggel

- char *getLevelFromFile(renderAssets *renderAssets)

  Kirajzolja a menüt a pályaválasztáshoz,majd a kiválasztott pálya elérési helyével visszatér

- void renderAnimateEntity(renderAssets *renderAsset, Level *level, SDL_Texture *texture, Coordinate newCoords, Coordinate oldCoords, Direction dir)

  Animálja a megadott karaktert

- void renderAnimateEntityMoveUp(renderAssets *renderAsset, SDL_Rect tile, LevelDimensions size, SDL_Texture *texture, Coordinate newCoords, Coordinate oldCoords)

  Animálja a megadott karakter felfelé haladását

- void renderAnimateEntityMoveDown(renderAssets *renderAsset, SDL_Rect tile, LevelDimensions size, SDL_Texture *texture, Coordinate newCoords, Coordinate oldCoords)

  Animálja a megadott karakter lefelé haladását

- void renderAnimateEntityMoveLeft(renderAssets *renderAsset, SDL_Rect tile, LevelDimensions size, SDL_Texture *texture, Coordinate newCoords, Coordinate oldCoords)

  Animálja a megadott karakter balra haladását

- void renderAnimateEntityMoveRight(renderAssets *renderAsset, SDL_Rect tile, LevelDimensions size, SDL_Texture *texture, Coordinate newCoords, Coordinate oldCoords)

  Animálja a megadott karakter jobbra haladását

- void renderHUD(renderAssets *renderAsset, Coordinate hudZero, Player *playerOne, Player *playerTwo, bool renderIce)

  Kirajzolja a HUD-ot

- void renderClearHUD(renderAssets *renderAsset, Coordinate hudZero)

  Clear-eli a HUD-ot

- void renderPlayerHUD(renderAssets *renderAsset, Coordinate start, Player* playerTwo, const char* palyerName)

  Egyjátékos adatait rajzolja ki a HUD-ra

- Level *initLevel(renderAssets *renderAsset)

  Inicializál egy pályát, először pálya választást kéri a felhasználótól, majd elvégzi a pálya felépítsését

- void insertNameToplist(renderAssets *renderAsset, Place **topListArray, int *placementNumber, int score)

  Bekéri egy játékos nevét, ha felkerült a toplistára és frissíti az Eredmények.txt tartalmát, majd megjeleníti a listát

IOput.c :

- MainMenuSelection getMainMenuInput(SDL_Event *eventMain, MainMenuSelection selection, renderAssets *renderAsset)

  A főmenüben kezeli az inputot, ha billentyűlenyomás visszatér a megfelelő értékkel, ellenben selection a kimenet

- Place **readTopList(int placementNumber)

  Eredmények.txt-ből feltölt Place strukturát, annyi darabot ammenyi a placementNumber bemenet

- char *remakeDynString(char newchar, char* string)

  Egy dinamikus stringhez hozzáfűz egy karaktert, majd a régit felszabadítja, és visszatér az új stringre mutató pointerrel

- char *removeDynStringLastChar(char *string)

  Egy dinamikus stringből töröl egy karaktert, majd a régit felszabadítja, és visszatér az új stringre mutató pointerrel

- char *duplicateDynString(const char* string)

  Duplikál egy dinamikusan foglalt tömböt

- int numFromString(char *string)

  String karaktereket integer számmá alakítja majd visszatér

- void freePlaceStructArray(Place **list, int size)

  Felszabadít egy Place strukturákból álló dinamikus tömböt

- void placeStructArraySelectionSort(Place **list, int size)

  Pont szerint csökkenő sorrendbe rak egy Place strukzúrákból álló tömböt

- void writeTopList(Place **list, int size)

  Eredmények.txt fájlba kiírja a toplistát

- Place **remakeDynPlaceStructArray(Place **oldArray, int size)

  Place tömbhöz hozzáad egy üres elemet , majd felszabadítja a régi listát, és felötli a megadott értékekkel

- char getSDLInputABC(SDL_Event *eventInput)

  Visszaadja a letütött betű billentyűt char formátumban, 0 ha nem betű a lenyomott gomb, 1 ha ENTER, 3 ha SHIFT, és ha egyik feltétel sem teljesül akkor 2-t ad vissza

- char **readLevel(const char* filePath, LevelDimensions *dim)

  filePath eléri útból betülti egy dinamikusan foglalt 2D karaktertömbe a txt tartalmát

- void free2DCharArray(char **array, int rowSize)

  Felszabadít egy "D dinamikusan foglalt tömböt

- TileType **mapConverter(char **array, LevelDimensions dim)

  2D karaktertömbből 2D TileType enumtömböt készít, megfeleően elkészíti a hátteret

- char* makeDynString(int size)

  size méretű dinamikus stringet készít

- void freeLevel(Level *level)

  Felszabdítja a level struktúra dinamukisan foglalt map 2D TileType tömbjét

- char charFromNum(int number)

  0-9 ig megadott number számot ad vissza char váltózóként

- windowData getWindowData(void)

  Konzolban kéri a felhasználót a felbontás és a teljes képernyős mód kiválasztására

- void renderConsoleP3Kmen(void)

  P3Kmen feliratot ír ki a console abalakba

- void renderConsoleResChooseMenu(void)

  Kiírja a felbontás választómenüt a console abalakba

- void renderConsoleFullscreenChooseMenu(void)

  Kiírja a teljesképernyős mód választómenüt a console abalakba

- bool getFullsreenFromConsole(void)

  Igaz ha a felhasználó a teljes képernyős módót választotta

- Resolution getResolutionFromConsole(void)

  Azzal az elemmel tér vissza, amely felbontást a felhasználó választotta

gamelogic.c :

- Player playerInit(renderAssets *renderAsset, Difficulty difficulty, TileType pacmanID, Level *level)

  Inicializál egy játékost

- void spawnPlayer(Level *level, Player *player)

  Lespawnol egy adott játékost a pályára

- void spawnMonster(Monster *monster, Level *level)

  Lespawnol egy adott szörnyet a pályára

- bool moveIsValidCheck(Level *level, Coordinate coordinate)

  Igazzal tér vissza ha a lépés engedélyezett vagyis ha nem fal, egyébként hamis

- bool isPointCheck(Level * level, Coordinate coordinate)

  Igazzal tér vissza ha a megadott elem pont

- bool isLittlePointCheck(Level * level, Coordinate coordinate)

  Igazzal tér vissza ha a megadott elem kis pont

- bool isPathCheck(Level * level, Coordinate coordinate)

  Igazzal tér vissza ha a megadott elem út

- bool isMonsterCheck(Level *level, Coordinate coordinate)

  Igazzal tér vissza ha a megadott elem szörny

- void sumOfAvalaiablePoints(Level* level)

  A level struktúra megfelelő pontkába összeszámolja az elérhető pontszámot

- void monsterInit(Level *level, Monster *monster, SDL_Texture* texture, TileType id)

  Szörny típus inicializálása

- double distanceFromAtoB(Coordinate A, Coordinate B)

  A és B pontok távolságával visszatér

- Coordinate *getPlayerCoords(Level *level)

  Megadja egy játékos koordinátáit

- int getPlayerNumber(Level *level)

  Megadja egy játékosok számát

- Coordinate getValidCoords(Level *level)

  Visszatér egy olyan koordinátával amire lehet lépni

- void playGame(renderAssets *renderAsset, Level* level, Player *playerOne, Player *playerTwo, Monsters* baddies)

  Egy játék játszása

- void keyBoardInputHandler(renderAssets *renderAsset, Level* level, SDL_Event *event, Player *playerOne, Player *playerTwo, Monsters *monsters)

  A bemenő billentyűzet imputot kezeli

- void playerMoveToNewDest(renderAssets *renderAsset, Level *level, Player *player, Direction dir, Monsters *monsters)

  Átrak egy játékost másik pontra

- Coordinate getCoordinateFromDirection(LevelDimensions size, Coordinate coords, Direction dir)

  Irányból megmondja a koordinátát

- void moveEntityToDest(renderAssets *renderAsset, Level *level, TileType id, Coordinate oldCoords, Coordinate newCoords, SDL_Texture *texture, Direction dir, bool isPlayer)

  Karaktert rak át a megadott pontra

- Direction getDirectionFromCoords(Coordinate newCoords, Coordinate oldCoords)

  Koordinátákból megadja az irányt

- bool gameEndCheck(Level* level)

  Igaz ha véget ért a játék

- Coordinate getHUDZeroPoint(Level *level)

  Visszatér a HUD kezdőkoordinátájával

- Uint32 timer(Uint32 ms, void *param)

  Megadott ms-bemenet alapján ms milisecundomonként SDL_USEREVENT usereventet rak az SDL event queue-ba | Használat: 1 mp-es időzítő az eventek időzítéséhez

- Uint32 timerMonsterAiKill(Uint32 ms, void *param)

  Megadott ms-bemenet alapján ms milisecundomonként SDL_USEREVENT + 1 usereventet rak az SDL event queue-ba | Használat: időzítő a függvényhez, ami a szörny játékos megevesésért felelős

- Uint32 timerMonsterAiMove(Uint32 ms, void *param)

  Megadott ms-bemenet alapján ms milisecundomonként SDL_USEREVENT + 2 usereventet rak az SDL event queue-ba | Használat: időzítő a függvényhez, ami a szörny mozgásáért felelős

- bool updateHUDCheck(int *playerPointsOld, int* playerSumHealtOld, Player *playerOne, Player *playerTwo)

  Igazzal tér vissza ha a megadott adatok frissültek.

- void clockHandler(TimeHandler *eventTimer)

  Kezel egy TimeHandler struktúrát, alap időegység: mp

- void spawnUsables(Level *level)

  lespawnolja a módosítókat

- void levelInitEventTimer(TimeHandler *eventTimer)

  Inicializál egy eventTimer strukturát

- void timedEventHandler(Level* level, Monsters *monsters, Player *playerOne, Player *playerTwo)

  Időzített események kezelése

- void initEventTimeAndCondition(eventTimeAndCondition *event)

  Inicializál egy eventTimeAndCondition strukturát

- void printDataToConsole(Level *level, Player *playerOne,Player* PlayerTwo, Monsters *baddies)

  kiírja az adatokat a consolba a játékról

- void bigPointsInit(Level* level)

  Elhelyezi a nagy pontokat a pályán véletlenszerűen

- MoveEnable moveAIPlayerKill(Direction dir, Coordinate coords, Level* level)

  Visszatér egy MoveEnable struktúrával aminek a tartalmát az határozza meg hogy,a szörny megehet egy játékost

- `void monsterAiKill(renderAssets *renderAsset, Level* level, Monster *monster, Player *playerOne, Player *playerTwo)`

  Vizsgálja hogy a szörny megehet-e egy játékost, ha igen, akkor végrehajta, vagyis megeszi

- `void monsterAiMove(renderAssets *renderAsset, Level *level, Monster *monster)`

  Mozgat egy szörnyet

- `void respawnPlayer(Monsters *monsters, Level *level, Player *player)`

  Újra spawnol egy játékost, a szörnyektől egy bizonyos távolságra

- `Coordinate getValidPathCoords(Level *level)`

  Visszér egy véletlenszerű koordinátával, ami path-re mutat

- `bool checkMovementValidDir(Level* level, Coordinate coords)`

  Igaz ha a megadott irányba a lépés megtehető

- `Coordinate getCoordsFromDir(Direction dir, Coordinate coords)`

  Megadott irányba visszter a megfelelő új koordinátával

- `bool preferredDirAiMove(renderAssets *renderAsset, Level *level, Monster *monster)`

  Igaz a Monster strukturúban a dir változó irányába a lépés megtehető

- `int getPointsFromDifficulty(Difficulty difficulty, int score)`

  A játék alatt szerzett pontokatt módósítja a nehézség függvényében

- `void freeMonsters(Monsters *monsters)`

  Felszabadít a Mosnters strujtúrát

- `Monsters *initMonsters(renderAssets *renderAsset, Level* level)`

  Inicializálja a szörnyeket

- `void playSFX(Mix_Chunk *sfx)`

  Ha sfx nem NULL, lejátsza a hangeffektet