Rittgasszer Ákos
Z8WK8D

Reversi

Programozói dokumentáció

A program használja az SDL2 libraryt amivel grafikus kirajzolást hajtom végre. Csináltam három header filet, a settings.h-t, a draw.h-t és a game.h-t. A settingsben vannak létrehozva az adatszerkezetek és azok használatát könnyítő függvények. A draw grafikus kirajzolást végrehajtó függvényeket tartalmazza, míg a game felel a játék zavartalan működéséért. Ezeknek megfelelően létezik settings.c, draw.c és game.c, valamint a main.c ami a fő vezérlést tartalmazza. A programhoz tartozik még egy save.txt állomány, amibe lehet elmenteni egy játékmenetet majd beolvasni azt. Ezeken kívül használja még a time.h headert, ezen kívül csak a szokásos headerek kerülnek includálásra.

A settings.h-ban van az adatszerkezet. Van egy game nevű structurám amiben benne van a játéktábla mérete (a szabályzatnak megfelelően 8), egy kétdimenziós dinamikusan kezelt tömbbe (board) ami Cell nevű stucturákból áll, valamint benne van még a játék módja és a következő játékos. A mód is és a játékos is egy-egy enum, a Mode lehet person és computer, a Player lehet black, white és none (az utóbbi itt nem kerül használatra). A Cell structurában benne van a cella két koordinátája és az, hogy mi van éppen azon a cellán (Player típusú). Van továbbá néhány függvény is a header fileba. Egy, ami egy game és két koordináta alapján visszaadja a kijelölt cella pointerét. Egy, ami megadja a következő játékost, egy, ami a játék módját, egy, ami az oszlopok számát és egy, ami a sorok számát. Illetve amelyik megmondja, hogy egy adott cella benne van-e a kétdimenziós tömbben. Végül van benne egy Delay függvény, ami adott ideig várakozik, emiatt csatolni kellet a time.h filet. Ez a header mindegyik másikhoz includeálva van. A settings.c ugyanezeknek a függvényeknek a kódját tartalmazza.

A draw.h-ban van egy windowData structura amiben benne van az ablak fejlécének szövege mérete és hogy egy cella mekkora. Van egy Point structura ami egy pontot határoz meg x és y koordináták alapján. Továbbá egy Color structura amiben színeket lehet eltárolni R, G, B és A paraméterekkel. Van egy függvény, ami magasság, szélesség, fejlécszöveg és cellaméret alapján létrehoz egy WindowData típusú változót. Egy, ami inicializálja az ablakot egy WindowData és egy SDL_Window és SDL_Renderer alapján. És egy amelyik megrajzolja az éppen aktuális állás (game) alapján a pályát. A draw.c tartalmazza ezeknek a kódjait, illetve segédfüggvényeket a kirajzoláshoz. Ilyen függvény a SetBackground ami megrajzolja a zöld háttérű négyzetrácsos képet úgy, hogy a széleken hagy egy félcellányi keretet. A DrawBorder függvény ezt a keretet rajzolja megy olyan színnel, ami a következő játékos színe. A DrawLine az előbbi függvényhez rajzol adott ponttól adott pontig megadott színű és vastagságú egyenest.

A game.h a játék menetét végrehajtó függvények helye. Ebben van az a függvény, ami lefoglalja a dinamikus a kétdimenziós Cellekből álló tömböt, illetve amelyik felszabadítja azt. A fileba írás és olvasás is itt található. A mentés úgy történik, hogy egy sorban van a következő játékos és a játék módja majd utána egy méretszer méretes területen ott van minden cella tulajdonosa. Az alap Step függvény, ami egy lépésért felel kap egy cellát, megkapja a game-et, és egy logikai változót, ami azt, mondja meg, hogy hajtsa-e végre a forgatásokat vagy ne. Végül visszatér azzal, hogy sikeres volt-e a lépés. A feltételként megnézi, hogy az adott cellának a lehetséges lépései között van-e olyan, ami forgatással jár (PossibleStep, visszatér a forgatható cellák számával) és ha forgatni akartunk akkor megnézi, hogy a következő játékos fog tudni rakni-e, ha igen akkor ellentétesre állítja a következő játékost, ha nem tud rakni akkor viszont marad és ugyanaz a játékos jön. A PossibleStep paramétere a game, az aktuális cella és egy logikai változó,

hogy végre kell-e hajtani a forgatást. A függvény, ha nincsen vége a játéknak és a megadott cella üres akkor meghívja azonos bemenő paraméterekkel az AllTurns függvényt majd visszatér annak értékével, ha nem akkor 0 a visszatérési értéke. Az AllTurns függvény meghívja minden irányra a Turn függvényt. Az irányokat egy xdir és egy ydir változó határozza meg, ha 0 akkor azon tengely mellet nincs elmozdulás, ha 1 akkor növekednek a koordináták értékei, ha -1 akkor csökkennek. Az AllTurns összeadja az összes lehetséges irányban a forgatható cellák számát és visszatér vele. A Turn függvény bemenő paramétere az irány (xdir és ydir) valamint ugyanazok, mint az AllTurnsnek. A Turn megnézi, hogy az adott cella kielégíti-e a szükséges követelményeket, ha igen akkor addig megy az adott irányban amíg nem talál az aktuális játékos színével azonos cellát vagy üres cellát. Ezután, ha nem üres cellánál állt meg visszafele megy végig és megszámolja hány cellán ment át és ha a bemenő turn változó true volt forgatja őket. Végül visszatér a számukkal.

A main.c-ben van a fő vezérlés. Itt létrehozom a játékot és beolvasom, hogy folytatni vagy újrakezdeni kell-e és hogy mi a mód. Létrehozom az ablakot és kirajzolom a pálya alapállását. Ezután jön a játék, A Play függvényben. Ebben egy while ciklus van amíg vége nincs a játéknak vagy amíg be nem zárjuk az ablakot. Az egér kattintás event segítségével megkapom a kattintott pixel koordinátáit, amit aztán elosztva a cella nagyságával megkapom, hogy melyik celláról van szó. Ezután erre meghívom a Step függvényt és kirajzolom a pályát a változásokkal együtt. Ezután, ha a játék módja ember elleni akkor vár egy újabb kattintásra.

Ha gép ellen játszunk akkor Ezután meghívom a Step függvényt úgy, hogy az átadott cella a BestTurn függvény visszatérési értéke. Ide beraktam egy egymásodperces várakozást, hogy jobban lehessen látnia történteket. Ezután ismét kirajzolom a táblát. Miután kiléptem a ciklusból meghívom az EndGame függvényt, ami megnézi a nyertest és közli azt a játékossal. Ezután, ha vége lett a játéknak resetelem a pályát és végül elmentem majd felszabadítom a foglalt memóriákat.