NHF – Specifikáció

A program neve:

Töltés-játék (Charge!)

A program célja:

Egy olyan játék létrehozása, melyben egy kilőtt pozitív töltésű töltést a felhasználó által lerakott különböző töltések segítségével egy megadott területre kell juttatni, melyet a játék pontoz, dicsőséglistát tart számon a legjobb eredményekről.

A program használata:

A program első megnyitásakor a kezdőképernyőn megjelenített menüből kattintással el lehet jutni a játék dicsőséglistáját, új játékot lehet kezdeni, folytatni lehet a már megkezdett játékot, valamint ki lehet lépni a programból.

Az első opció megjeleníti a dicsőséglistát, melyet egy fájlban tárolunk (HallOfFame.txt). Ez a fájl minden sorában tartalmaz egy pontértéket, majd tabulátorral elválasztva a játékos nevét. (A pontszámítás formulája majd a létrehozás folyamatában pontosul, de alapvetően beleszámít a lerakott töltések száma, és a próbálkozások száma.)

Az új játékot választva a játékos az első pályán találja magát (legalább 5 pályát tartalmaz a végleges program), ahol tetszőleges mennyiségű pozitív illetve negatív töltést rakhat le kattintással a pálya különböző pontjaira. Ezek a fizika törvényei alapján taszítják, illetve vonzzák az eredeti részecskét.

A pályák ezen felül tartalmaznak falakat, melyek antianyagból vannak, ütközés pillanatában megsemmisül a részecske. Ezeket a Levels.txt fájl tartalmazza.

A részecske az indító gomb megnyomásakor elkezd mozogni a rá ható erőknek megfelelően. Ekkor a felhasználó nem tud módosításokat végezni a pályán, csak nézni az eredményt. Minden egyes nyomás ezen a gombon hozzájárul a próbálkozások számához, ami beleszámít a játékos végső pontjába. A játék a pálya tejesítésekor számolja a játékos pontját, az utolsó pálya végén frissíti a dicsőséglistát. Ebben a módban lehetőségünk van menteni, és visszalépni a menübe.

A megkezdett játék folytatásakor értelemszerűen egy elmentett állást tölthetünk be. A pont számításakor irreleváns, hányszor töltöttük be a végigjátszást, illetve hogy mennyi idő telt el a megkezdés és a befejezés között.

A kilépés gomb megnyomására a program leáll

Felhasználói felület:

A program grafikus megjelenésű, 1280*720 pixeles nem átméretezhető ablakkal rendelkezik.. A kezdőképernyőn a játék neve (Charge!) szerepel, és a négy alapvető opció egymás alatt, középre zárva szerepel. A játékon belül egy sárga kör jelenti a kezdő részecskét, a lerakhatóakat piros (+), illetve kék (-) körök. A falak színe szürke, a kijelölt célmező áttetszően piros. A képernyő tetején megtalálható a menü, ahol a felhasználó látja a pálya megkezdésekori pontját, nevét, töltéseket, amelyeket kiválasztva rakhat le belőlük, mentés, kilépés és indítás gombot. Az indításkor megjelenik egy időzítő, amely méri, mennyi idő alatt ért a töltés a célba.

NHF – Programozói Dokumentáció

Követelmények:

A program a C szabványos könyvtárain felül az SDL könyvtárait és a Windows LATINWD betűtípusát használja, így ezek megléte szükséges – értelemszerűen a program az utóbbi miatt Windows operációs rendszerhez kötött.

Adatszerkezetek:

```
/** Tetszőleges vektor    */
typedef struct Vector{
    double vx;          //x irányú komponens
    double vy;          //y irányú komponens
}Vector;

/** Pályán található részecske    */
typedef struct Particle{
    SDL_Point pos;      //A részecske helye
    int radius;         //A részecske sugara
    int charge;         //A részecske töltése
    int mass;           //A részecske tömege
    Vector vel;         //A részecske sebessége
    Vector acc;         //A részecske gyorsulása
}Particle;

/** Listaelem, amely a pályára lerakott részecskéket tartalmazza */
typedef struct ParticleList{
    Particle part;                  //A lista eleme; részecske
    struct ParticleList *next;      //A következő elemre mutató pointer
}ParticleList;

/** Listaelem, amely a pályán levõ falat tartalmazza */
typedef struct WallList{
    SDL_Rect wall;                  //A lista eleme; egy fal
    struct WallList *next;          //A következő elemre mutató pointer
}WallList;
```

```
/** Tetszőleges szöveg adatait tárolja(megjelenítés helye, szöveg színe, betűtípusa, a szöveg maga)  */
typedef struct Szoveg{
    SDL_Rect hova;              //A szöveg helye
    SDL_Color szin;             //A szöveg színe
    TTF_Font *font;             //A betűtípus
    char str[200];              //A szöveg
}Szoveg;

/** A dicsőséglista eleme    */
typedef struct Champion{
    int Score;                  //A győztes pontja
    char Name[51];              //A győztes neve
}Champion;
```

```
/** A pálya összes adatát tartalmazó struktúra    */
typedef struct Level{
    ParticleList *parts;        //Részecskék listája
    Particle part;              //A játékban mozgó részecske
    int lvlNr;                  //A szint száma
    int partNr;                 //A szinten lerakott töltések száma
    SDL_Point orPos;            //A részecske eredeti helye
    Vector orVel;               //A részecske eredeti sebessége
    WallList *Walls;            //A szinten lévő falak listája
    SDL_Rect goal;              //A célmező
}Level;

/** A játékos összes adatát (név és pont) tartalmazó struktúra    */
typedef struct Player{          //A jobb olvashatóságért külön a Champion-től
    char Name[51];              //A játékos neve
    int Score;                  //A játékos megszerzett pontja
}Player;

/** A játék éppeni állását tartalmazó struktúra  */
typedef struct Gamestate{
    Player player;              //A játékos
    Level level;                //Az aktuális szint
    int TryNr;                  //Az eddigi próbálkozások száma

}Gamestate;
```

Függvények, modulok szerint:

megjelenites.c:

void sdl_init(const char *felirat, int szeles, int magas, SDL_Window **pwindow, SDL_Renderer **prenderer)

Inicializálja a grafikus ablakot, amelyen keresztül a felhasználó a programmal kommunikál.

void szoveg_kiir_kozep(SDL_Renderer *renderer, Szoveg szov, SDL_Rect *gombok)

Kiírja a megkapott szoveget a képernyőre középre igazítva, és hozzárendel egy gombot.

void szoveg_kiir(SDL_Renderer *renderer, Szoveg szov, SDL_Rect *gombok)

Kiírja a megkapott szoveget a képernyőre, és hozzárendel egy gombot.

void menu(SDL_Renderer *renderer, SDL_Rect *gombok)

Megjeleníti a menüt

bool Button_Pressed(SDL_Event ev, SDL_Rect gomb, SDL_Point p)

Leteszteli, hogy megynyomtak-e egy megadott gombot.

void HallOfFame_kiir(Champion *Hall, const int HoF_Nr, SDL_Renderer *renderer, SDL_Rect *gomb)

Megjeleníti a dicsõséglista elemeit.

void Valaszt(char szo[], char kar, char szo1[], char szo2[])

6. heti laborban megírt stringszétválasztó függvény.

void HallOfFame(SDL_Renderer *renderer)

A dicsõséglista betöltéséért és a menüpont végrehajtásáért felelõs függvény.

bool input_text(char *dest, size_t hossz, SDL_Rect teglalap, SDL_Color hatter, SDL_Color szoveg, TTF_Font *font, SDL_Renderer *renderer)

Beolvas egy szoveget a billentyuzetrol. A rajzolashoz hasznalt font es a megjelenito az utolso parameterek. Az elso a tomb, ahova a beolvasott szoveg kerul. A masodik a maximális hossz, ami beolvasható. A visszateresi erteke logikai igaz, ha sikerult a beolvasas.

void GetName(Gamestate *state, SDL_Renderer *renderer)

Inicializálja a változókat, hogy a játékos neve beolvasható legyen. Ezután elkészíti az olvasáshoz a képernyőt, és beolvassa a játékos nevét.

void GameCntrllr(SDL_Renderer *renderer, SDL_Rect *gombok, Gamestate state, int charge)

Kirajzolja a képernyõre az aktuális játékállás szerinti vezérlőpultot.

void GameLevelDraw(SDL_Renderer *renderer, Gamestate *state)

Megjeleníti a képernyőre a pálya jelenlegi állását.

void InHoF(Player player)

Megnézi, hogy az elért pontszámmal a játékos bekerül-e a dicsőséglistába, majd ha igen, berakja.

void Won(SDL_Renderer *renderer, Gamestate state)

Megjeleníti a győztes feliratot, frissíti a dicsőséglistát.

jatek.c:

int HoFCount()

Visszatér a dicsőséglistában tárolt személyek számával.

bool TestCollision(Particle part, SDL_Rect Wall)

Megnézi, hogy egy részecske ütközött-e egy fallal, vagy elérte-e a célmezõt

Vector Force(Particle p1, Particle p2)

Kiszámolja két részecske közötti elektromágneses erõt.

SDL_Point NextPos(SDL_Point p, Vector v, int t)

Kiszámolja a következõ idõegység lejárta utáni helyét egy részecskének.

Level GetLevel(FILE *fp)

Kiolvassa a pálya adatait egy fájlból.

void SaveGame(Gamestate state)

Elemnti a jelenlegi játékállást egy fájlba. A fájl relatív helye: \SaveGame\current_save.txt

bool LoadGame(Gamestate *state)

Betölti a jelenlegi játékállást egy fájlból. A fájl relatív helye: \SaveGame\current_save.txt

Uint32 idozit(Uint32 ms, void *param)

Ez a fuggveny hivodik meg az idozito altal.

Vector SumForce(Particle part, ParticleList *first)

Összegzi a listában összes részecske által kifejtett erőt a részecskére.

void GamePlay(SDL_Renderer *renderer, Gamestate *state, SDL_Rect *gombok, SDL_Rect field, int charge, bool *win)

Az indítás gomb utáni történéseket vezérli: A részecske mozgását, ütközéseket, pontszámítást.

void GameEventLoop(SDL_Renderer *renderer, Gamestate *state, SDL_Rect *gombok, int *charge, bool *win)

A játéktörzs event loopja, az átláthatóság kedvéért.

void Game(SDL_Renderer *renderer, Gamestate *state)

A láték futását intézi

lista.c:

ParticleList *addPart(ParticleList *first, Particle data)

Hozzáad egy meglévõ részecskelistához egy megadott részecskét.

WallList *addWall(WallList *first, SDL_Rect data)

Hozzáad egy meglévõ fallistához egy megadott falat.

Particle newPart(int x, int y, int charge)

Létrehoz egy új részecskét.

void freePart(ParticleList **first)

Felszabadítja a meglévõ részecskelista összes elemét, a pointerjét NULL-ra állítja.

void freeWall(WallList **first)

Felszabadítja a meglévõ fallista összes elemét, a pointerjét NULL-ra állítja.

NHF – Felhasználói Dokumentáció

A program használata:

A játék indításakor megjelenő menüben a szövegekre kattintva lehet navigálni a program funkciói között. Az Exit gomb kilép a játékból, a View Hall of Fame gombbal megtekinthetjük a dicsőséglistát, majd a Back gombbal visszajutunk a menübe. A Continue Game gomb betölti a már mentett játékállást (ha van). A New Game opcióval elkezdődik a játék. A felhasználó beírhatja a nevét, amely egyből elmentődik (maximum 50 karakter, de a megjelenítés miatt kevesebb az ideális. ű, ő, és egyéb speciális karaktereket nem jelenít meg).Enter lenyomásával elmentődik a név, és az első pályán találjuk magunkat (Összesen 5 pálya van). A játék célja a sárga töltést eljuttatni a piros mezőbe, töltések lerakásával. Ezt a pályára kattintva tehetjük meg. A kontrollpanelen látható dobozra kattintva megváltoztathatjuk a lerakandó részecske töltését. A Reset gomb az összes töltést eltűnteti. A Save gomb elmenti a jelenlegi játékállást, míg a Quit gomb kilép a menübe. A játékot a START! gomb indítja, amikoris nem lehet már több töltést lerakni, elindul a részecske, és vagy nekiütközik valaminek, vsgy eléri a célt. Utóbbi esetén a következő pályán találjuk magunkat, az előbbi pedig azt eredményezi, hogy előről kezdhetjük a szintet. Az 5. pálya sikereres befejezése egy gratulációs üzenetet okoz, majd visszaléptet a menübe, ahol az előbb leírtak szerint megtekinthetjük, hogy bekerültünk-e a dicsőséglistába.