Kiindulásként a programozási nyelvet kívántam megválasztani, olyat, ami a feladathoz megfelelő és az iparban elterjedt nyelv a mai nap is.

Egy széleskörben elterjedt, az számos iparágban\* jelenleg is használatos nyelvet válaszottam így, amiben lehetséges objektum orientált szemléletet követni.

Az algoritmusok hatékony működése mellett szükségesnek tartottam egy gyors nyelvet is, hogy ezen tényező biztosan ne legyen hátráltató tényező.

Ezért döntöttem a C++\* nyelv mellett.

A megjelenítéshez kellett választanom egy grafikus könyvtárat, a Simple DirectMedia Layer 2-t (SDL2) válaszottam.

az alapvető funkciókat ellátja, egyszerűbb geometriai alakzatok kirajzolására képes grafikus könyvtár.

Működése egyszerű, hatékonysága különösebben ennél a feladatnál nem számít, mert nem szükséges komoly látványos grafikát megvalósítanom.

A feladathoz szükséges adatok értelmezését nagyban segítő reprezentáláshoz bőven elegendő eszköztárral rendelkezik.

Továbbá a felhasználói bemeneteket, mint az egér mozgatása és a vele történő kattintás és görgetés, továbbá a billenytűzet minden gombját is kezeli.

Ezért megfelelő a feladathoz.

Ezt követően, előnyeben részesítve a nyelvet, elkezdtem keresni a témában munkákat, cikkeket, hogy a program tervezésnél minél több problémára tudjak figyelni.

Kiindulásként szükséges volt keresnem olyan már létező megoldásokat, ahol az útvonalkereséssel már évek óta foglalkoznak.

A játékiparban már évtizedek óta foglalkoznak azzal, hogy különböző egyedek, emberek vagy objektumok a gép által vezérelve eljussanak A pontból B pontba.

Egy nagyon régi példát említve már több, mint negyven éve alkották meg az első PAC-MAN\*-t, amelyben a játékost a játéktéren szellemek próbálták elkapni.

Ebben a példában a szellemekből négy volt és mindegyiknek saját logikája volt.\*

Az útvonalkeresést Dijkstra algoritmus segítségével is valósíthatták.\* A teljesség kedvéért kifejtem az algoritmust.

Az algoritmust a legrövidebb út problémára megoldás. Edsger W.Dijkstra (1930 - 2002) holland matematikustól származik.

Az algoritmushoz szükséges egy élsúlyozott gráf és csak abban az esetben működik helyesen, ha minden él súlya nemnegatív.

Jelen példánkban nincs erre precedens, így alkalmas megoldás az útvonalkeresésre, továbbá összefüggő is a gráfunk így kevesebb problémát kell vizsgálnia.

Ugyanis a játéktér tekinthető egy négyzetrács hálónak, és ha jobban megnézzük, akkor ezen háló szabad mezőiből tudunk csoportokat alkotni, amelyeket élekként és csúcsokként gráfba tudunk szervezni.

Az algoritmus mindig a legrövidebb útvonalat adja vissza a gráfban, és futásideje is kedvező.

Az algoritmus egy gráfot használ keresési tereként, és két a csúcsát kezdő és végpontként.

Kezdő csúcs minden szomszédos csúcsát egy értékkel látja el.

Az csúcsba vezető élek súlyát veszi a csúcsok értékének.

Ezt követően kiválasztja azt a csúcsot, amelynek az értéke a legkisebb, és az ő szomszédjainak az értékét úgy adja meg, hogy nem csak a csúcshoz vezető él súlyát veszi figyelembe,

hanem a azt is, hogy magának a csúcsnak mekkora az értéke. Azaz a csúcshoz vezető út éleinek súlyának mennyi az összege a kezdőpontból.

Fontos, hogy ha annak a csúcsnak, melynek értéked ada, már van értéke, és az kisebb vagy egyenlő, akkor nem bírálja felül, és a kisebb értékét meghagyja.

Ellenkező esetben a kifejtett csúcson keresztüli út élek súlyainak összességét adja neki értékül.

Ez az eset azt jelenti, hogy vagy elsőként adott értéket a csúcsnak, azaz fedezte fel a gárfban, vagy talált egy rövidebb utat a kezdőpontból a csúcsig.

A csúcsok kifejtése, és az algoritmus futása addig tart, amikor a végpont csúcsát fejtené ki. Ekkor ugyanis a végpont csúcsának értéke adja meg, a legrövidebb utat a két csúcs között.

Azért fontos, hogy ne akkor álljon le az algoritmus, amikor először felfedezi az végpont csúcsát, mert még utána lehet, hogy talál egy rövidebb utat a végpontig.

Viszont, mikor a végpont csúcsát fejti ki az algoritmus, akkor az azt vonja magával, hogy más kifejtésre váró csúcs értéke legalább annyi, mint a végpont csúcsa.

Ekkor mivel nincsen negatív él, ezért a kifejtendő csúcsokból már biztosan nem lehet rövidebb utat találni a végpontba, így a feladatát az algoritmus elvégezte, és leállhat.

Fontos, hogy ezen algoritmus használata során felfedezett csúcsokhoz el kell tárolni az értékén kívül a kíszámításának módját is, valamilyen módon.

Különben egyetlen számértéket visszakapva a legrövidebb utat nem tudnánk meghatározni, csak a hosszát az útnak.

Ezért fontos, például a felfedezett csúcsokhoz eltárolni az értékükön kívűl azt is, hogy melyik csúcsból lett kifejtve az értékük.

Ebben az esetben a végpontból kiindulva meg kell vizsgálni azt a csúcsot, amiből ki lett fejtve a legkisebb értéke.

És azt a csúcsot is, amelyből az ki lett fejtve, és így tovább egészen a kezdőpont csúcsáig.

Bizonyítás nélkül, de igaz, hogy ezen út így előáll a visszafele haladva megvizsgált csúcsokból és a legrövidebb is lesz a két pont között.

Ezen algoritmus nagyon közel áll egy mai napon is, a játékiparban standard algoritmushoz, az A\* (kiejtve a csillag) algoritmushoz.

Ennek az algoritmusnak több átdolgozott verzióját is használtam a programomban végül.

Az Dijkstra algoritmus publikálását követően tizenkettő évvel később publikálták az A\*-ot először, az Dijkstra algoritmus kiegészítéseként.

Az algoritmus abban volt több, és a gyakorlatban a későbbiekben sokkal hatékonyabb, hogy a csúcsokhoz rendelt értéket új heurisztikával adta meg.

Az A\*-ot akkor lehet használni, ha létezik olyan költség becslést végző függvény, amely a végpont csúcsáig tetszőleg csúcsból képes olyan távolság becslést adni,

amely sosem lehet több, mint a tényleges út hossza, abból a csúcsból.

Ekkor ugyanis a heurisztika a következő.

A csúcs értékének ne az odáig vezető út éleinek súlyát adjuk meg, hanem külön adjuk hozzá ennek a becslés végző függvény értékét a csúra nézve.

Az onnan kifejtett csúcs értékét nem az előző csúcs értéke és az összekötő él súlya adja majd,

hanem az előző csúcs érétkének a becslést végző függvény értékének hozzáadását megelőző érték, az összekötő él súlya és a becslő függvény értéke a csúcsra adja majd.

Az algoritmus továbbra is egy csúcs kifejtése során minden szomszédos csúcsot kifejt, viszont akkor áll le, amikor először felfedezi a csúcsot.

A heurisztika és az algoritmus végének módosítása hatékonyabbá teszi az útvonalkeresést nagyságrendekkel.

Gyakori megoldás a költség függvények a két csúcs távolsága a térben. Erre lehetőség van olyan esetekben, amikor a gráf csúcsai valamilyen teret írnak le.

Ilyen tér például lehet egy térképen a városok

PAC-MAN példáját véve az A\* nem használható könnyen, mert a játéktér két pontja között nem lehet könnyen helyes becslést végezni.

Ugyanis a viízszintesen középen lévő átjáró a játéktér egyik oldalát összeköti a másik oldalával. Ekkor ami a képernyőn egy egység távolságnak tűnik, az igazából az átjáró okán lehet nulla is.

Ennek okán a becslő függvényt megfelelően kéne megválasztani ahhoz, hog

A feladatom az, hogy alkossak egy olyan programot, amely képes a járókelők mozgásának, azaz a gyalogos közlekedés leszimulálására.

Kezdetben a feladatom, hogy a problémafelvetéssel és az irodalomkutatással kezdjem. A kettőt szükséges együtt végezni, ugyanis a már meglévő, mások által írt munkákban a problémakör nehézségére gyakran kitérnek.

Ennek okán olyan feladatokat és problémákat volt feladatom találni a témán belül, amikkel a feladatomat kibővíthetem.

Ezt követően folytassam az irodalomkutatást, hogy a kibővített problémakörre találjak már létező megoldásokat, vagy csak segítséget ahhoz, hogy ki tudjak indulni valamiből.

Mások munkáit olvasva a program egyes részproblémáinak megoldására próbáljak egyre több féle megközelítést találni, és ezeket mindenképpen hasonlítsam majd össze, és jussak valami következtetésre.

Az így keletkezett megoldásokat fel tudjam használni majd a programomban és végül egy bizonyos szempontból egy optimális megoldással tudjak előállni.

Az szerzett ismeretek alapján ezt követően feladatom, hogy specifikáljam a program elvárt működését, majd tervezzem is meg.

Tervezés közben gondoljam végig, hogy mi szükséges a program elkészítéséhez és a különböző tervezői döntéshelyzetekben idoklással hozzak lehetőleg helyes döntéseket.

Tervezés közben teszteljem a megoldásomat gyakran, és ha lehet a problémát osszam többrészre, hogy elkülönítve a tesztelés csak egy-egy egységre történjen egyszere.

Ezzel segítsem azt, hogy a program felépítése legyen egységekre bontva, és megfelelően szervezve.

Ezt mind a működőképességért, az áttekinthetőségéért, és a jövőbeli továbbfejlesztési lehetőség fenntartása érdekében tegyem meg.

Majd a program elkészülését követően vizsgáljam meg a működését, és telszteljem helyességét a gyakorlatban.

A szimuláció paramétereit a valóságnak megfelelően választva vessem össze, hogy az eddigi ismereteinkkel összeegyeztethető az eredmény.

Ha nem az elvárt eredmény kapom, vizgsáljam át esetleges hibák után kutatva a programom, vagy indokoljam meg, hogy mi okozhatja a eltérést.

Program készítése során feladatom már létező algoritmusok részletesebb megismerése, és számomra specifikus algoritmusok kifejlesztése és létrehozása.

Ezen algoritmusok nem csak gyorsak legyenek, hanem hatékonyak is, és lehetőleg hasonlítsak össze több megoldást is.

Eznen algoritmusokhoz feladatom olyan adatmodellt alkotni, amellyel könnyen lehet dolgozni, és a felépítése logikus.

A programot feladatom úgy létrehozni, hogy lehessen paraméterezni, például ne csak egy előre betáplált területtel, alaprajzzal dolgozzon.

Lehessen új alaprajzokat megadni a programban, és lehessen változtatni az emberek főbb paramétereit az adott környezetre jellemző értékekre.

Mindehhez egy olyan grafikus felületet alkossak, amely alkalmas az adatok olyan megjelenítésére, hogy könnyen értelmezhetőek legyenek, és a program működése szemmel látható legyen.

Az élet alapvető és szervez részét képzi az, hogy tudunk tájékozódni, és közlekedni.

Tudatában kell lenni az emberek tulajdonságaival és viselkedésükkel, továbbá a normákkal ahhoz, hogy a többi emberrel együtt lehessen élni.

Alkalmazkodni kell hozzájuk, és nyíltan sosem kimondva, együtt kell dolgozni velük. Ezek nélük az együtt élés lehetetlen lenne.

Remek példa erre a közlekedés.

Fontos, hogy legyenek egymásra tekintettel az emberek, ezekben segítenek a normák. Fontos ugyan úgy, hogy a helyzetfelismerő képességük is helyes legyen, ehhez figyelniük kell, és alapvető ismeretekre van szükségük.

Ahhoz, hogy a helyzetfelimserése helyes legyen az embereknek, tudniuk kell olyan alapvető dolgokat, mint hogy hol vannak, kik vannak körülötte és ők merre tartanak.

Merre fele tudnak haladni, korlátok, falak, lépcsők és sok más terepobjektum milyen lehetőségekre korlátozzák a továbbhaladásukat.

Ahhoz, hogy a céljukhoz eljussanak éppen sétálniuk kell-e, mert nem sietnek, állniuk kell-e, mert piros a lámpa, vagy liftben vagy villamoson vannak, esetleg futniuk kell-e, hogy elérjék a buszt,

mind mind ismerniuk kell a környezetüket, és saját maguk tulajdonságait, és korlátait.

Nem tudunk tetszőlegesen gyorsan haladni, mert képtelenek vagyunk egy adott sebességnél gyorsabban futni.

Nem tudunk minden kanyart belátni, minden sarkon túl ismerni a környezetet, mert fal vagy más eltakarja.

Nem tudunk mindig beszállni a liftbe, mert nincs hely már számunkra, vagy tovább sem tudunk haladni, mert elfáradtunk.

Végtelenül összetett feladat az, hogy egy ember eljusson A pontból B pontba, miközben ezt a hétköznapokban maguk az emberek nem is érzékelik különösebben komplex problémának.

Pedig számos tény és ok alakítja mozgásukat és viselkedésüket.

Informatikában ezen problémák megmaradnak komplex feladatoknak. Számos tényt és okot figyelembe kell venni ahhoz, hogy a valósághoz hasonlót sikerüljön lemodellezni az emberek mozgásával kapcsolatban.

Hogy az emberek mozgása végül természetesnek hasson, és a valógágot minél jobban megközelítse.

Ehhez temérdeknyi kutatás és adat szükséges, hogy akár csak kezdetleges megoldást, modellt készítsünk, ami leírná az emberek - a közlekedésben a járókelők - mozgását.

Ha az emberek számára egy pillantás is elegendő, hogy felmérjék a közlekedésbe a szituációt, miért szükséges bevonni az informatikát, amelyben ezen probléma megoldása nehezebb.

Ugyanis gyakran nincs lehetősége az emberekne egy pillantást vetni a környezetre, esetnenként nincs még meg a környezet sem, vagy automatizálásra van szükség.

Azért fontos foglalkozni azzal, hogy helyesen lehessen szimuláni a járókelők mozgását, hogy az emberek élete kellemesebb, és komfortosabb legyen.

Rengeteg tervezés előz meg egy épület megépítését, egy villamos összeszerelését, vagy egy kereszteződés átalakítását.

Azért szükséges a tervezéshez sok idő, hogy a lehető legjobb lehessen a végeredmény, és minél kevesebb problémája legyen.

Az épület esetében nem csak az a fontos, hogy minél magasabb vagy épp impozánsabb legyen, hanem többek között az is, hogy a legapróbbnak tűnő dolgok rendben legyenek.

Például ne nyíljanak egymásba az ajtók, a lépcsőfokok mélységének és magasságának aránya megfelelő legyen, ne legyenek a szobák és a folyosók túl szűkek, vagy épp túl elnagyoltak.

Hogy a tervezés végére minden a rendeltetésének megfelelő méretű legyen, ahogy legyen ergonómikus elrendezésű is és az embereknek természetesnek ható és komfortos is.

Azzal, hogy a szimuláció a járókelők mozgásáról helyes, azzal a terveket ellenőrizni lehet, hogy tényleg elfér-e ott annyi ember.

Hogy tényleg elegendő-e harminc másodperc egy kereszteződésnél, hogy a forgalomszámlálást követően megbecsült maximális száz ember átérjen a zebrán.

Vagy, hogy egy lakást az ember várhatóan kellőképpen ki fogja-e használni, a bejárhatósága lehetséges-e, és ezek mind másokkal való együtt élés esetén is igazak maradnak-e.

Vagy, hogy egy épületet tűzriadó esetén el tudja-e hagyni adott időn belül az épületben tartózkodók mindegyike maximális kapacitás esetén is.

Ezzel mind lehet, és egyes esetekben kell is a tervezés során foglalkozni.

Kell számolni azzal, hogy a tervezett objektum kiknek készül, pédául betegeknek, idősebbeknek vagy szélesebb embereknek.

A lépcsők az idősebbek számára fontos kérdés, míg a szélesebb emberek számára meg a tömegközlekedésen a székek szélessége fontos.

Kórházak és egyéb közintézmények esetén az akadálymentes közlekedést kötelező megoldani.

Egy olyan programon dolgoztam, ami a járókelők mozgásának szimulációjára alkalmas, főképp, de nem szükségesen csak zárt terekben.

Használható arra például, hogy egy adott épületnek egy adott emeletéről szerezzünk információkat, hogy mennyire használható,

és a tervezett maximális kapacitás esetén hogy zajlana le egy tűzriadó, vagy csak a hétköznapokban hol okoz fennakadást egy szűkület, például az ajtók vagy a sarkok.

Ehhez szükségem volt az emberekre általánosságban elmondható tulajdonságok összegyűjésére,

a különböző paramétereiknek az átlagos értékeiknek meghatározására, és a közöttük lévő kapcsolatok megismerésére.

Gondolok paraméterek alatt a sebességükre, a korukra, a magasságukra, a szélességükre, a privátszféra kiterjedtségére, és a tömegben történő mozgás okozta bizonytalanságból adódó egyénre gyakorolt hatásra is, többek között.

Kapcsolatok alatt például az embertömeg sűrűsége és az egyén mozgásának sebessége közötti kapcsolatra gondolok.

?

A program elkészítése során szükségem volt egy teszt környezetre, amiben vizsgálhatom, hogy a programom helyesen működik-e.

A programomat az egyetemi polgárokra terveztem és ehhez az egyetem I épületének negyedik emeletét modelleztem le.