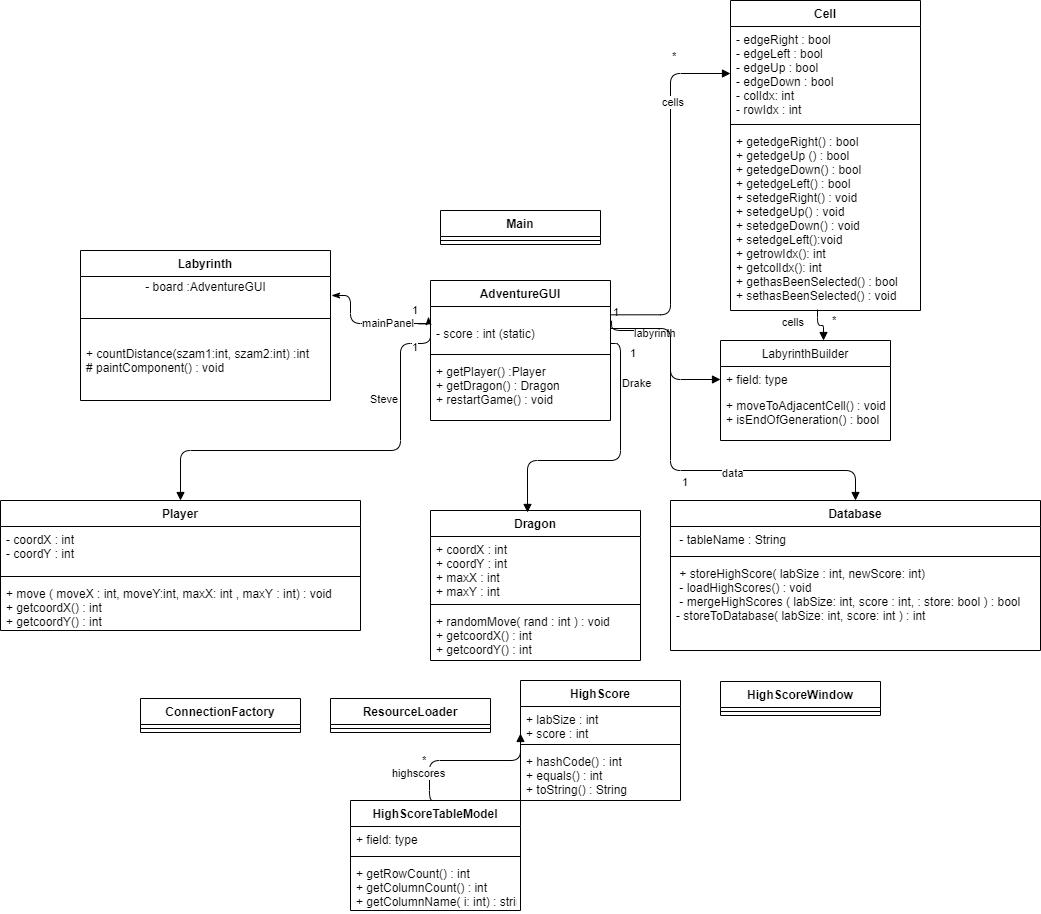
Lukács Dávid István, idu27k   
programozási technológia

3.beadandó

A feladat leírása  
  
  
Készítsünk programot, amellyel egy labirintusból való kijutást játszhatunk. A játékos a labirintus bal alsó sarkában kezd, és a feladata, hogy minél előbb eljusson a jobb felső sarokba úgy, hogy négy irányba (balra, jobbra, fel, vagy le) mozoghat, és elkerüli a labirintus sárkányát. Minden labirintusban van több kijutási útvonal. A sárkány egy véletlenszerű kezdőpozícióból indulva folyamatosan bolyong a pályán úgy, hogy elindul valamilyen irányba, és ha falnak ütközik, akkor elfordul egy véletlenszerűen kiválasztott másik irányba. Ha a sárkány a játékossal szomszédos területre jut, akkor a játékos meghal. Mivel azonban a labirintusban sötét van, a játékos mindig csak 3 sugarú körben látja a labirintus felépítését, távolabb nem. Tartsuk számon, hogy a játékos mennyi labirintuson keresztül jutott túl és amennyiben elveszti az életét, mentsük el az adatbázisba az eredményét. Egy menüpontban legyen lehetőségünk a 10 legjobb eredménnyel rendelkező játékost megtekinteni, az elért pontszámukkal, továbbá lehessen bármikor új játékot indítani egy másik menüből. Ügyeljünk arra, hogy a játékos, vagy a sárkány ne falon kezdjenek.

Osztálydiagram

mETÓDUSOk

## adventuregui::adventuregui

Grafikus UI konstruktora,, melyben meghívom a labirintusgenerálást, létrehozzuk az összes UI elemet, generáljuk a játékost és a sárkányt.

## adventuregui::restartgame

A játék újraindító metódusa, újra inicalizálja az adattagokat, törli az előző ablakot, majd megnyit egy újat.

## adventuregui::keyadapter

A billentyűlenyomáshoz kötött eseménykezelőnk. Ebben az esetben a sárkányt egy véletlenszerű irányba lépteti, mindenképpen olyan irányba ahol nincsen fal, a játékosunkat pedig a lenyomott billentyűnek megfelelő irányba mozdítjuk el, de csak akkor, ha az adott irányba történő elmozdulást fal nem akadályozza. Ezután minden billentyűlenyomásnál ellenőrizzük, hogy a játékost elpusztította-e a sárkány, illetve ha ez nem történt meg, akkor a játékosunk nyert-e. Ha ezek nem történtek meg, frissítjük a grafikus felület elemeit.

## cell::selectimage

Képkiválasztó metódus, az éleket jelentő boolokat vizsgáljuk. Ha egy bool true, akkor van edge (fal) , ha false, akkor nincs. Egy hosszú elágazás során valamelyikben mindenképp megáll a metódus, és az annak megfelelő képfájlt tölti be a ResourceLoader segédosztállyal.

## dragon:: randommove

A sárkány mozgató metódusa, végülis egy hosszú, irányválasztó elágazás a random szám függvényében

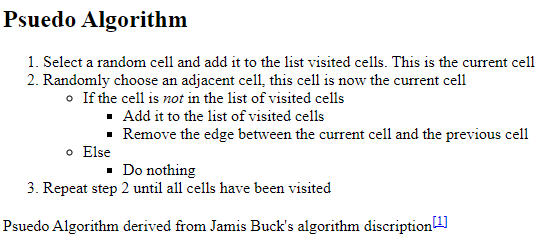
## labyrinthbuilder::movetoadjacentcell

Aldous-Broder Algoritmusnak a szomszédos mezőt kiválasztó rész implementációja.

## labyrinthbuilder::isendofgeneration

Eddig megy az algoritmusunk, tehát amíg nincs minden elem legalább egyszer megjelölve. Ellenőrzi, hogy minden cella meg van-e jelölve.

## labyrinthbuilder::labyrinthbuilder

Tkp. egy Aldous-Broder algoritmus .   
A lényege, hogy választunk egy véletlenszerű cellát, majd elindulunk egy véletlenszerű irányba, és ha az a cella még nem volt kiválasztva, akkor kiválasztjuk, és a két cella közti élt töröljük. Ezt addig csináljuk, amíg nincs minden cella kiválasztva.

## labyrinth:paintcomponent

Grafikai rajzoló metódus, itt hívjuk meg a képkiválasztó függvényket, és helyezzük őket a megfelelő helyre az ablakon belül.

# tesztelés

## Játék elindítása, többszöri újraindítás

A játékot már a menüben, pályaválasztás előtt is többször újraindítjuk, majd utána vizsgáljuk, hogy megfelelően újra inicializálódnak-e az adattagjaink, illetve hogy például a pontszámunk visszaáll-e, újra töltődik-e az adatbázis, és ha a legnagyobb pontszámunk az aktuális, akkor frissíti e azt.

## egyszeri végigjátszás

Ebben a tesztesetben azt teszteljük, hogy ha először elindítjuk a játékot, tudunk-e nyerni, módosul e a pontszámunk, illetve elindul e egy új pálya.

## másodszori végigjátszás

Itt azt teszteljük, hogy egy első megnyert játék után a második alkalommal nem-e borul fel valami, megmarad e a pontszámunk, működik-e, változik-e a pálya, máshova kerül-e a sárkány, ugyanoda a játékos, frissül-e a toplista.

## más pályaméretek

Itt azt vizsgáljuk, hogy megfelelően működik-e a különböző pályaméreteknél a labirintusgenerálás. Ez jelenleg a kódba még nincs implementálva, hogy menüben változtatni lehessen. de az algoritmus helyességét jól tesztelhetjük ezzel.

## a bűvös sárkány

Ellenőrizzük azt, hogy ki tudunk e kapni a játékban, kellőképpen elpusztulunk, amikor a sárkány a közelünkbe ér, illetve hogy például falon keresztül nem-e képes megölni minket a sárkány.

## a falak azok falak(?)

Ellenőrizzük azt, hogy a játékos, illetve a sárkány át tud-e menni a falakon, ki tud-e menni a pályáról.

## Minden út rómába vezet

A labirintusgeneráló algoritmust vizsgáljuk, hogy minden esetben van-e kiútja a labirintusnak.