Pikachu’s Revenge

<https://github.com/richardfoltin/PikachusRevenge>  
ELTE IK - Programozási technológia 1.  
3.feladatsor/egyéni feladat, 1. feladat alapján

*Foltin Csaba Richárd – I37M02*

# Feladatleírás

## Közös követelmények

A megvalósításnak, felhasználóbarátnak és könnyen kezelhetőnek kell lennie. Törekedni kell az objektumorientált megoldásra, de nem kötelező a többrétegű architektúra alkalmazása.

A megjelenítéshez első sorban elemi grafikát kell használni. Az így kirajzolt ’sprite’-ok közül a játékoshoz tartozót billentyűzet segítségével lehessen mozgatni a jelenleg is standard (WASD) billentyűzet kiosztásnak megfelelően. Egyéb funkciókhoz egérhez kapcsolódó esemény vezérlőket is implementálhattok.

Amennyiben nem algoritmussal generáltatod a játékteret, úgy legalább 10 előre definiált játékteret készíts különböző fájlokban eltárolva. Ügyelj arra, hogy mind az algoritmussal generált játékok esetén, illetve az előre definiált esetekben is végig játszható legyen az adott terület.

Minden feladathoz tartozik egy időzítő, mely a játék kezdetétől eltelt időt mutatja.

A dokumentációnak tartalmaznia kell a választott feladat leírását, elemzését, a program szerkezetének leírását (UML osztálydiagrammal), egy implementációs fejezetet a kiválasztott játék szempontjából és/vagy az általad érdekesebbnek gondolt algoritmusok leírásával. (Például: pálya generáláshoz implementált algoritmus.), valamint az eseményeseménykezelő párosításokat és a tevékenység rövid leírását.

A feladatleírás a minimális követelményeket tartalmazza. A játékok tetszőlegesen bővíthetők.

## Maci Laci feladat alapján

A meséből jól ismert Maci Laci bőrébe bújva a Yellowstone Nemzeti Park megmászhatatlan hegyei és fái között szeretnénk begyűjteni az összes rendelkezésre álló piknik kosarat. Az átjárhatatlan akadályok mellett Yogi élelem szerzését vadőrök nehezítik, akik vízszintesen vagy függőlegesen járőröznek a parkban. Amennyiben Yogi egy egység távolságon belül a vadőr látószögébe kerül, úgy elveszít egy élet pontot. (Az egység meghatározása rád van bízva, de legalább a Yogi sprite-od szélessége legyen.) Ha a 3 élet pontja még nem fogyott el, úgy a park bejáratához kerül, ahonnan indult.

A kalandozás során, számon tartjuk, hogy hány piknik kosarat sikerült összegyűjtenie Lacinak. Amennyiben egy pályán sikerül összegyűjteni az összes kosarat, úgy töltsünk be, vagy generáljunk egy új játékteret. Abban az esetben, ha elveszítjük a 3 élet pontunkat, úgy jelenjen meg egy felugró ablak, melyben a nevüket megadva el tudják menteni az aktuális eredményüket az adatbázisba. Legyen egy menüpont, ahol a 10 legjobb eredménnyel rendelkező játékost lehet megtekinteni, az elért pontszámukkal, továbbá lehessen bármikor új játékot indítani egy másik menüből.

## További egyéni célok

* A karakterek képe tényleg mozgás hatását keltse, ne csak egy statikus kép legyen
* A pályák egy spritesheetből legyenek felépítve
* A karakterek ne csak függőlegesen és vízszintesen mozogjanak, hanem koordinátarendszer alapján

# Feladat Elemzése

Első lépésként a játék témájának meghatározása volt a fő cél. Mivel az Interneten Pokémon témában rengetegsok spritesheetet és karaktersprite-okat lehet találni ezért adta magát a téma. A játékban Pikachut lehet majd irányítani, hogy a pályákon elszórt pokélabdákból kiszabadítsa az elkapott pokémonokat. Eközben a pályákon mászkáló pokémontrainerek őt próbálják elkapni, amint a látószögükbe kerül.

A pályákat egy külső spritesheet editor programmal szerkesszük. A program generál egy file-t amely a spritesheeten található 16x16-os sprite-okhoz azonosítót rendel, és még esetleg egyéb paramétereket is megadhatunk nekik. Továbbá minden pályához tartozik egy-egy file ami leírja layer-enként hogy melyik azonosítójú sprite hova kerül. A pályát leíró file tartalmazza továbbá az NPC-k haladási útvonalát és paramétereit.

A játék JFrame ablakokat fog használni a program megjelenítésére. Az ablakon található egyéb elemek (Főmenü, pályák, információs panelek) cserélődni fognak.

A játék egy JScrollFrame-ben zajlik ahová betöltjük a pályát. A játék logikáját egy külön Model objektumban kezeljük. A program egy végtelenített ciklusban 40ms-onként újraszámolja a pozíciókat és újrarajzolja az ablakot. A játékot az irány billentyűkkel lehet irányítani ezért figyeljük a lenyomott billentyűket és ezekből 8 irány (4 alap + 2 egyszerre nyomott billentyű) alapján mozgatja Pikachut. Amennyiben egy a pályán található pokélabdára lép egy játékos, onnan kikel egy random pokémon. Az ellenfelek a pályán járőröznek a megadott útvonal szerint. Amennyiben a játékos a ellenfelek előtti 135°-ban tartózkodik a megadott távolságon belül akkor először megállnak – hagynak egy kis időt a játékosnak elmenekülni – de utána elkapják és a játékos visszekerül a kezdőpozícióba. A kiszabadított pokémonok megmaradnak, de egy élte elveszik. A pályát a minden pálya végén található tábla mellett SPACE-t nyomva lehet továbbvinni, amennyiben elegendő pokélabdát talált meg a játékos.

Azt hogy a játékos ne tudjon áthaladni a tereptárgyakon (házak, fák, víz, hegyek, …) úgy oldjuk meg, hogy a spritesheethez tartozó információs file-ban az collisiont okozó sprite-oknak paramétert adunk. A játékos mozgatásával egy hozzá tartozó láthatatlan négyzetet is mozgatunk, és ha ez egy tereptárgyon akarna áthaladni, akkor nem mozgatjuk oda a játékost. Mivel a pályák több layerből is állhatnak ezért megoldható hogy egy másik layeren a tereptárgyon áthaladható dolog legyen (híd, lépcső, …). A layereken készítünk mindig egy layert, amit majd mindig a játékosok felett rajzolunk ki.

A játéktér felett egy címkén megjelenítjük a játékos életerejét, az pályán található kiszabadított és ki nem szabadított pokémonokat és egy időt, mely a pálya kezdetétől eltelt időt méri.

A játékosok segítségére egy alsó információs címkét tervezünk, mely segítő és információs üzenteket küld a játékos számára.

A játékhoz tartozik egy menüsor is mely a szokásos mentés, betöltés, újraindításon kívül tartalmaz még egy a játék óráját szüneteltető és újraindító menüt. Továbbá egy dicsőséglistát.

# Használt Eszközök

* Tiled (<https://www.mapeditor.org/>) – Könnyen kezelhető közösségi fejlesztésű spritesheet editor, mely tartalmaz a generált file-ok beolvasásához használható java library-t is.
* Pokémon Insurgence (<https://p-insurgence.com/>) – Közösségi fejlesztésű pokémon játék, ahonnan a játékban használt képek és spriteshet származik
* Maze Creator (<http://www.mazegenerator.net/>) – A játék 9. pályáján található labirintus legeneráláshoz
* GitHub (<https://github.com/richardfoltin/PikachusRevenge>) – A játék fejlesztést folyamatosan publikáltam a GitHubra
* GenMyModel (<https://app.genmymodel.com/>) – UML creator GitHub alapján

# Felhasználói Esetek

A felhasználók a következőket tudják tenni a játékelindítása után:

1. Főmenüben
   1. Új játék indítása -> utána kiválasztása a nehézségi foknak
   2. Játék betöltése file-ból
   3. Játék betöltése adatbázisból
   4. Dicsőségtábla megjelenítése
   5. Az ablak bezárása
2. A játékban (ugyanazt mint a főmenüben kiegészítve a következőkkel)
   1. Pikachu irányítása WASD vagy a nyíl billentyűkkel
   2. SPACE lenyomásával interakcióba lépés Pikachu közelében lévő tereptárgyakkal (ha lehetséges)
   3. A játék elmentése file-ba
   4. A játék elmentése adatbázisba
   5. Visszalépés a főmenübe
   6. A játék lepause-olása és újra aktiválása
   7. Az adott pálya újrakezdése
   8. Egy már teljesített pályára visszalépés
   9. Az elkezdett pályák statisztikáinak megjelenítése
   10. A segítség ablak előhívása
   11. A megtalált pokémonokhoz kapcsolódó internetes leírás előhívása

# Program Szerkezete

A programban az összes használt osztály és a hozzájuk tartozó bonyolultabb metódusok ki vannak egészítve a működésüket leíró Javadoc-kal. Itt az érdekesebb, kihívásokat jelentő eseteket mutatom csak be

## A pálya felépítése

## Két lenyomott iránygomb egyszerre érzékelése

## Pokemonok megpróbálják kikerülni az akadályokat

## NPC-k különböző állapotai

## Mikor dobnak az NPC-k?

## Elrejtett adatbázis id kinyerése a betöltési táblából

# Osztálydiagram

Mellékelt uml.png

# Tesztelési terv

A játék tesztelését két féleképpen végeztem; white box (JUnit tesztekkel) a játék iránytásáért, pozíciókért és koordinátákért felelős metódusokat, és black box (a játék használatával) a teljes játék működését teszteltem.

## Black box:

1. Főmenü
   1. játék indítása – SIKERES
   2. játék betöltése file-ból - SIKERES
   3. hibaüzenet hibás file esetén - SIKERES
   4. játék betöltése adatbázisból - SIKERES
   5. hibaüzenet hibás adatbáziskapcsolat esetén betöltéskor – SIKERES
   6. dicsőségtábla betöltése – SIKERES
   7. hibaüzenet hibás adatbáziskapcsolat esetén a dicsőségtáblakor - SIKERES
2. Játék
   1. Játékos
      1. Pikachu mind a 8 irányba megy WASD és nyilak haszálatával – SIKERES
      2. Pikachu nem át az akadályokon (fák, víz, hegyek, házak, kövek) – SIKERES
      3. Pikachu átmegy a vizen ha van rajta híd – SIKERES
      4. Pikachu átmegy a hegyen ha van rajta lépcső – SIKERES
      5. Pikachu eltűnik a magas tárgyak mögött – SIKERES
      6. a megfelelő animáció jelenik meg az adott mozgás esetén - SIKERES
      7. amennyiben Pikachut elkapják, visszakerül a kezdő pozícióba és egy életet veszít – SIKERES
      8. a pálya végén található táblával a következő pályára lehet továbbmenni - SIKERES
   2. NPC
      1. a pálya file-jában megadott NPC töltődik be a megadott paraméterekkel - SIKERES
      2. az NPC-k a megadott útvonalon mozognak és megfelelő pontokon várakoznak - SIKERES
      3. az NPC-k felett felkiáltójel jelenik meg ha meglátják Pikachut – SIKERES
      4. az NPC-k megállnak ha meglátják Pikachut – SIKERES
      5. az NPC-k eldobják a labdájukat ha már felkészültek, vagy ha Pikachu túl közel került – SIKERES
   3. Pokémonok
      1. amennyiben Pikachu rámegy egy pokélabdára kiszabadul a pokémon és megjelenik a felső címkén – SIKERES
      2. a kiszabadított pokémonok libasorban követik Pikachut – SIKERES
      3. a kiszabadított pokémonok ugyanúgy nem mennek át tereptárgyakon, csak ha Pikachu túl messze van – SIKERES
3. Menüsor
   1. File
      1. új játék indítása - SIKERES
      2. játék betöltése file-ból – SIKERES
      3. játék mentése file-ba – SIKERES
      4. játék mentése a betöltött file felülírásával – SIKERES
      5. játék betöltése adatbázisból – SIKERES
      6. játék mentése adatbázisba – SIKERES
      7. játék mentése adatbázisba felülírással – SIKERES
      8. visszalépés a főmenübe megerősítőablakkal – SIKERES
      9. kilépés megerősítőablakkal – SIKERES
   2. Game
      1. a pálya újrakezdése – SIKERES
      2. pálya újrakezdésekor feltöltődik az élet a pálya betöltésekori értékre – SIKERES
      3. pálya újrakezdésekor Pikachu visszakerül a kindulási pontra – SIKERES
      4. pálya újrakezdésekor a kiszabadított pokémonok újra labdába kerülnek – SIKERES
      5. pálya újrakezdésekor az NPC a kindulási helyükre kerülnek – SIKERES
      6. pálya újrakezdésekor az óra lenullázódik – SIKERES
      7. szintlépés másik pályára ha lehetséges (1-10) – SIKERES
      8. szintlépés csak már teljesített pályára lehet – SIKERES
      9. pause/resume – SIKERES
      10. dicsőségtábal megjelenítése – SIKERES
      11. az adott játék összesítő statisztikájának megjelenítése – SIKERES
      12. a segítség megjelenítése – SIKERES
      13. felugró ablakok esetén a játék szünetel és bezáráskor visszaáll – SIKERES
   3. Pokédex
      1. Csak a kiszabadított pokémonok láthatóak benne – SIKERES
      2. menüpontra kattintáskor megjelnik a pokémon html oldala – SIKERES

## White box

1. Position
   1. distanceFrom(Position d)
   2. movePosition(Direction d, double speed)
2. TilePosition
   1. tileCenter(TilePosition tpos)
   2. tileCoordFromMapCoord(double coord)
   3. fromMapPosition(Position p, int level)
3. KeyPressHandler
   1. lenyomunk egy gombot megfelelő-e az irány?
   2. lenyomunk egy másik gombot megfelelő-e a komponált irány?
   3. felengedünk egy gombot, akkor újra megfelelő-e az irány?
4. Direction
   1. getDirection(double x, double y)
   2. getDirection(Position from, Position to)
   3. getSecondDirection(Position from, Position to)
   4. isInDirectionOfSight(Direction from, Direction to)
   5. directionAngleStart(Direction d)