

Snake Dokumentáció

Hőgyes Krisztián (D85S3M)





User manual:

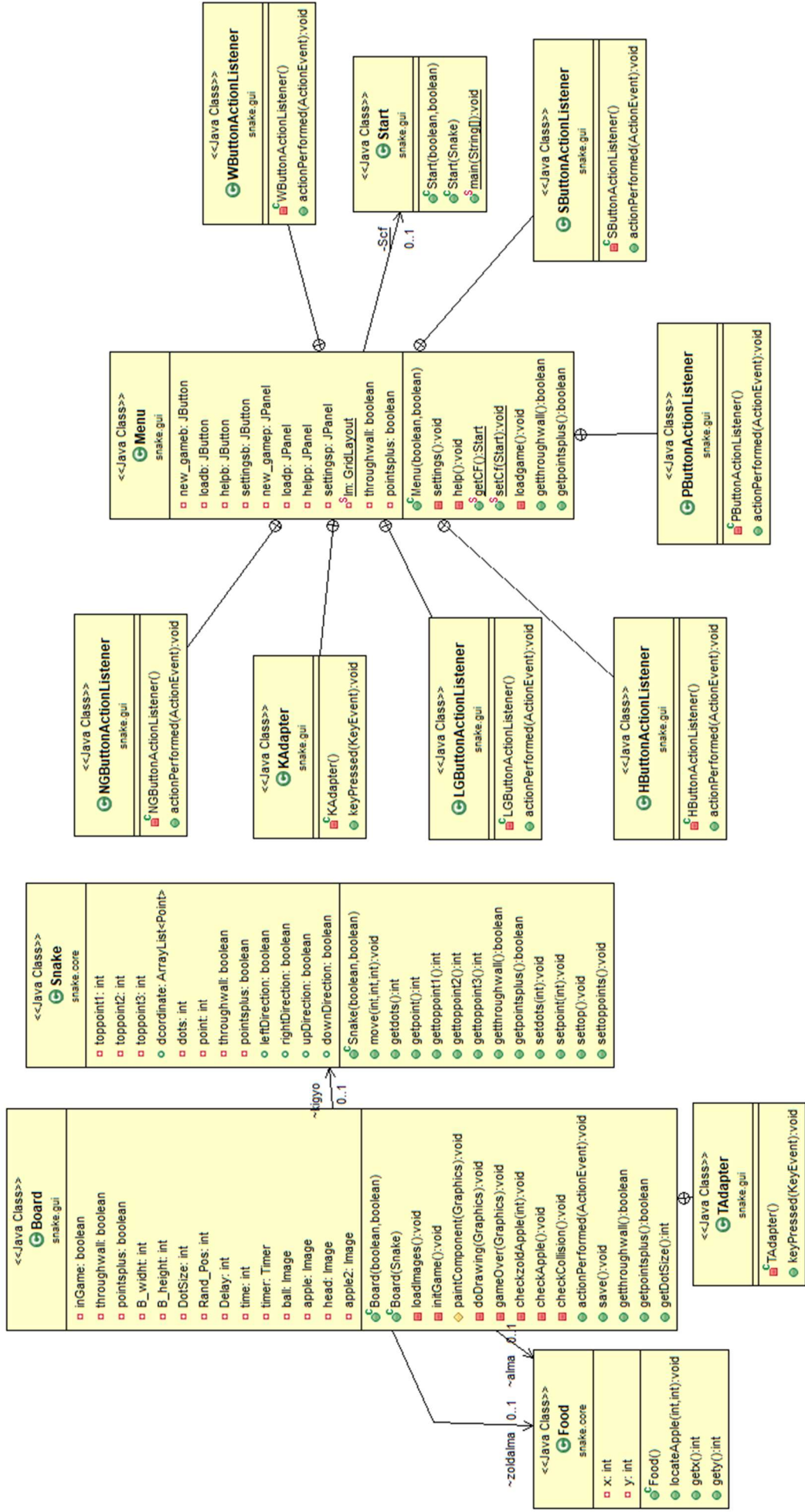
Snake egy játék, melynek lényege, hogy minél több pontot szerezz, azáltal, hogy a kígyóval felszeded az almákat. Kis nehézsége, hogy minden egyes megevett almával, nő a kígyó, ezáltal nehezítve a mozgást. Program indulásakor egy ablak jelenik meg a képernyő közepén, mely egy menüt tartalmaz. Menü részei:

- Új játék
- Betöltés
- Segítség
- Beállítások

Új játékot választva, értelemszerűen egy új játék indul, mindig ugyanaz a kígyó kezdőpontja, de az alma helyzete mindig random, így indulásnál is. Betölteni, csak akkor tudunk, hogyha előtte már mentettünk is (menteni játék közben tudunk az „s” billentyű lenyomásával), ekkor, amit legutoljára elmentettünk, azt fogja betölteni (csak egy pillanatnyi helyzetet lehet elmenteni, az van felülírva mindig).

Segítség menüpontban rövidítve le van írva, hogy mi a lényege a játéknak, illetve az irányítás. Beállításokban pedig lehet engedélyezni, hogy a kígyó át tudjon-e menni a falon (ekkor szemközti falon jön kiugyanott) illetve legyen-e plusz pontot érő zöld alma, ami a kígyó méretét nem növeli, de ha sikerül felszedni, akkor sokkal több pontot is szerezhethetünk. (Ahogy megjelenik 4096 pontot ér, de minden egyes lépéssel már csak a gyökét éri, azaz első lépés után 2048, második lépés után 1024 és így tovább, míg el nem tűnik. 12 lépésig él a lehetőség, és 5 almánként jelenik meg.)

A menüt egérrel lehet vezérelni, míg játék közben a kígyót a     billentyűkkel lehet irányítani, illetve az „s” betűt lenyomva lehet menteni, de attól függ a játék megy tovább, ha nem lépünk ki. Menüben az Esc billentyűvel lehet visszalépni.



Eclipse bővítmény által generált osztálydiagramm látható a képen, de a szabványos jelöléstől eltérően a teli rombusz(kompozíció) egy körnek van jelölve, melyben benne egy kereszt. A piros négyzet jelöli, hogy private, zöld, hogy public, illetve sárga a protected.

Osztályok leírása:

Start:

A program elindításáért felelős, tartalmazza a main osztályt, illetve két konstruktorral is rendelkezik, és mindkettő egy JFrame-t hoz létre, azaz egy új ablak keretet.

Menü:

Létrehoz egy JFrame-t majd JPanelekkel tölti fel, ez az osztály valósítja meg a kezdő menüt, azaz ez felelős a játék beállításáért, illetve meghívásáért is. A játék beállításáért, elindításáért, felelős, amellet, hogy egy alapmenüként szolgál.

Board:

A Start által létrehozott JFrame-t ez osztály tölti fel JPanelel, felelős a játék lebonyolításáért, azaz megjelenítéséért, és lefolytatásáért a kígyó osztály segítségével, ugyanis innen hívódik meg a kígyó mozgatása, de ez rajzolja minden egyes lépés után újra az egész pályát, az alma(k) és kígyó helyzetét. Felelős a játék közbeni irányításért.

Snake:

A kígyó tulajdonságait valósítja meg, ez tárolja a legjobb és aktuális eredményeket, hogy merre mozog a kígyó, s milyen nagy. A játék beállításainak megtartásáért, illetve a kígyóért felelős.

Food:

Az alma koordinátáinak megtartásáért, illetve kreálásáért felelős.

Start:

+Start(throughwall1: boolean, pointsplus1: boolean):

Az osztály konstruktora, mely két bool értéket kap, ezek felelősek az alapbeállítások miatt, azaz hogy van-e zöldalma, és átlehet-e menni a falon. Ezenkívül megvalósít egy JFrame-t, melyhez Board osztályt ad hozzá (JPanel), elnevezi az ablakot, láthatóvá és átmertethetetlenné, és középre teszi.

+Start(seged: Snake):

Az osztály másik konstruktora, mely a játék betöltése miatt jött létre. JFrame-t valósít meg, melyhez Board osztályt ad hozzá (JPanel), elnevezi az ablakot, láthatóvá és átmertethetetlenné, és középre teszi.

+main(args: String[]) :void

Program belépési pontja.

Menü:

-new_gameb, -loadb, -helpb, -settingsb: JButton a nyomógombok tárolója.

- new_gamep, -loadp, -help, -settingsp: JPanel a nyomógombok helyeiért felelősek.

-Scf:Start játék indítását teszi lehetővé.

-lm: GridLayout azablak felosztásáért felelős.

-throughwall: boolean a falon való átmenés „kapcsolója”.

-pointsplus: boolean a zöldalma lehetőségének „kapcsolója”

+Menu(throughwall1: boolean, pointsplus1: boolean):

Osztály konstruktora, létrehoz egy ablakot (JFrame) és ki is tölti azt, ezért 4 panelt és négy gombot jelenít meg. Minden gombot rendel egy nevet is, illetve egy eseményérzékelőt is, ezáltal megvalósítva, a vezérelhetőséget. NGButtonActionListener új játékot indít el. LGButtonActionListener betölti a játékot. HButtonActionListener a segítség menüpontot mutatja meg. SButtonActionListener program beállítását teszi lehetővé. KAdapter felelős, a menüben való visszalépésért.

-help():void

A játék rövid leírását teszi lehetővé, azáltal, hogy az eddigi használt JPaneleket eldobja, és hozzáad egy újat egy JTextArea-val, melyre azért esett a választás, mivel ez képes több sort is megjeleníteni. ezt a setLineWrap változójával lehet elérni.

-settings():void

Alap beállításokat ennek a függvénynek hála tudjuk elvégezni, az eddigi panelekből kettőt csinál, ahhoz hozzáadva egy gombot, mely igazából egy kétállású kapcsolót valósít meg a WButtonActionListener és a PButtonActionListener segítségével. (WButtonActionListener beállításokon belül a falak átjárhatóságának a kapcsolóját nyomja át. PButtonActionListener a zöldalma jelenlétét dönti el.)

+getCF():Start

Getter, Start osztályt ad vissza.

+setCF(cf : Start): void

Új Startot hoz létre, ezáltal elindítva a játékot.

-loadgame():void

Játék betöltését teszi lehetővé, azáltal hogy beolvasson egy fájlból, és az alapján hoz létre egy Snake példányt.

+getthroughwall():boolean

Visszaadja, hogy át lehet-e menni a falon.

+getpointsplus(): boolean

Visszaadja, hogy lesz-e zöldalma.

Board:

-inGame: boolean él-e a kígyó.

-throughwall: boolean a falon való átmenés „kapcsolója”.

-pointsplus: boolean a zöldalma lehetőségének „kapcsolója”

-B_widht: int az ablak szélességét tárolja.

-B_height: int az ablak magasságát tárolja.

-Rand_pos: int számolt érték, a szélesség, illetve magasságot elosztva a DotSize-vel kapjuk meg, és majd az almák random letételében segít.

-DotSize: int egy képkocka nagyságát jelzi. (betöltött képek pixel mérete)

-Delay:int késleltetés, ezzel érhető el, hogy lassabb legyen a kígyó mozgása.

-time: int a zöldalma életének idejét mutatja lépésekben mérve.

-timer: Timer időzítő, segít a játék folytonosságában.

kigyo: Snake kígyó osztály példánya.

alma, zoldalma: Food almák példánya.

-ball -apple -head -apple2: Image a betöltendő képeknek változója. Ezek segítségével rajzoljuk ki a kígyót és az almákat.

+Board(throughwall1:boolean, pointsplus1: boolean):

Konstruktor mely két „kapcsolót” kap és a Start által kreált JFrame-t tölti fel, és jeleníti meg segédfüggvényei által.

+Board(seged: Snake):

Konstruktor, mely egy Snake osztálybeli példányt kap, a saját Snake osztályú példányát ezzel teszi egyenlővé, ezáltal megvalósítva a betöltést. A Start által kreált JFrame-t tölti fel, és jeleníti meg segédfüggvényei által.

-loadImages(): void

Betölti a program számára használatos képeket, amivel lesz megjelenítve a kígyó, alma, zöldalma.

-initGame(): void

Kezdfelállást létrehozza, és elindítja a játékot.

#paintComponent(g: Graphics): void

Minden egyes lépés után kirajzoltatja az általa létrehozott „vászonra” a pálya aktuális állapotát.

-doDrawing(g: Graphics): void

A betöltött képeket rajzolja ki a „vászonra” a pálya állapotának megfelelően.

-gameOver(g: Graphics): void

Játék elvesztését teszi láthatóvá.

-checkzoldApple(pont: int):void

Ha a kígyó hossza 5-el osztva maradéktalan, akkor letesz egyzöldalmát, annak értéket ad, és ellenőrzi, hogy a kígyó megette-e, illetve csökkenti a zöldalma idejét. Ha a kígyó megette, akkor a kígyó pontjához hozzáadja a saját pontértékét.

-checkApple(): void

Ellenőrzi, hogy megette-e a kígyó az almát, ha igen, akkor lerakat egy újat a pályára, illetve a kígyónak a pontját növeli 8-al.

-checkCollision(): void:

Vizsgálja, hogy élünk-e, s ha meghaltunk volna, akkor egy boolean változó megváltoztatásával jelzi, hogy vége a játéknak.

-actionPerformed(e: ActionEvent): void

Ez a függvény teszi lehetővé a játék folytonosságát azáltal, hogy az ellenőrző függvényeket meghívja, illetve újrarajzolat.

+save(): void

Fájlba írja a játék aktuális állását.

+getthroughwall():boolean

Visszaadja, hogy át lehet-e menni a falon.

+getpointsplus(): boolean

Visszaadja, hogy lesz-e zöldalma.

Class TAdapter: a Board-on belül

+keyPressed(e: KeyEvent): void

Kígyó irányítását teszi lehetővé, illetve a mentést, azáltal, hogy a leütött billentyűket figyeli, és azalapján változtatja a kígyó irányát, vagy meghívja a fájlba írás függvényét.

Snake:

-toppoints1, -toppoint2, -toppoint3 : int, legjobb három eredmény tárolására van.

-dots: int a kígyó hosszát tárolja.

-point: int a kígyó pontját tartalmazza.

+dcoordinate: ArrayList<Point> Kígyó helyzetét tárolja, annyi elem van a listában, amennyi a dots értéke.

-throughwall: boolean a falon való átmenés „kapcsolója”.

-pointsplus: boolean a zöldalma lehetőségének „kapcsolója”

+leftDirection +rightDirection +upDirection +downDirection :boolean kígyó mozgásának irányát tárolják.

+Snake(throughwall1: boolean, pointsplus1: boolean):

Snake osztály konstruktora, a két „kapcsolót” megkapja, ezáltal tudva, hogy lesznek-e zöldalmák, illetve át tud-e menni a falon. A kígyó kúszásának iránya nincs eldöntve, hogy majd a játékos döntse el, azzal elindítva ténylegesen a játékot

.+move(Dotsize: int, B_widht: int, B_height: int):void

Kígyó mozgásáért felelős, annak megfelelően, hogy melyik irány van megadva igaznak, arra tolja el egyet a kígyót.

+getdots(): int

A kígyó hosszával tér vissza.

+getpoint(): int

Az elért ponttal tér vissza.

+gettoppoint1():int +gettoppoint2():int +gettoppoint3():int

A legjobb 3 eredményt adják vissza.

+setdots(i: int):void

A kígyó hosszát állítja be azáltal, hogy az eddigihez hozzáadja a kapott értéket.

+settop():void

A legjobb három eredményt írja meg.

+settoppoints(): void

Fájlból beolvassa a legjobb 3 eredményt.

+getthroughwall():boolean

Visszaadja, hogy át lehet-e menni a falon.

+getpointsplus(): boolean

Visszaadja, hogy lesz-e zöldalma.

Food:

-y, -x : int koordináta tárolására van.

+locateApple(Rand_pos:int, Dotsize: int):void

a Rand_pos azért van, hogy pályán belül bárhova tehesünk almát, a DotSize pedig megadja az alma méretét, így a függvény leteszi random helyre az almát.

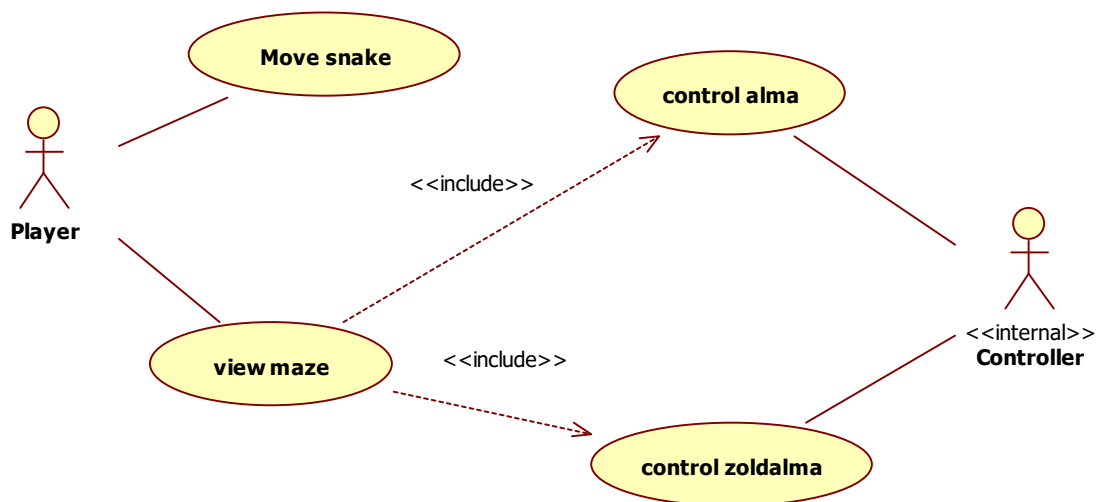
+getx(): int

Visszaadja x értékét.

+gety(): int

Visszaadja y értékét.

Use-case:



Cím	Move snake
Leírás	A játékos a kígyót irányítja a pályán.
Aktorok	Player
Főforgatókönyv	1. A játékos a kígyót felfel, lefelé, jobbra, vagy balra irányítja.
Alternatív forgatókönyv	1.A.1 Ha a kígyó megeszik egy almát pontot kap.
Alternatív forgatókönyv	1.A.2 Ha a kígyó megeszik egy almát nő.
Alternatív forgatókönyv	1.B.1 Ha a kígyó megeszik egy zöldalmát pontot kap.

Alternatív forgatókönyv	1.C.1 Ha a kígyó önmagának meg meghal.
Alternatív forgatókönyv	1.C.1.A.1 Ha a kígyó meghal végea játéknak.

Cím	view maze
Leírás	Játékos megtekinti a pályát.
Aktorok	Player
Főforgatókönyv	<ol style="list-style-type: none"> 1. A rendszer kirajzolja a pálya aktuális állapotát. 2. A játékos megtekinti a pálya aktuális állapotát.

Cím	control alma
Leírás	Alma van a pályán
Aktorok	Controller
Főforgatókönyv	1.A Mindig van egyalma a pályán
Alternatív forgatókönyv	1.B a kígyó megeszi az almát.

Cím	control zoldalma
Leírás	Van zoldalma a pályán
Aktorok	Controller
Főforgatókönyv	1.A Bizonyos időközönként megjelenik a pályán.
Alternatív forgatókönyv	1.B a kígyó megeszi a zöldalmát.

A program egy fájlba egy Snake osztálypéldányt ír ki, mely tárolja a kígyó helyzetét, a méretét, irányát, legjobb három és aktuális pontot. Ugyanezt is olvassa be.