Programozás alapjai 3.

Házi feladat

Zergi Máté-BKZ85U

Platformer játék

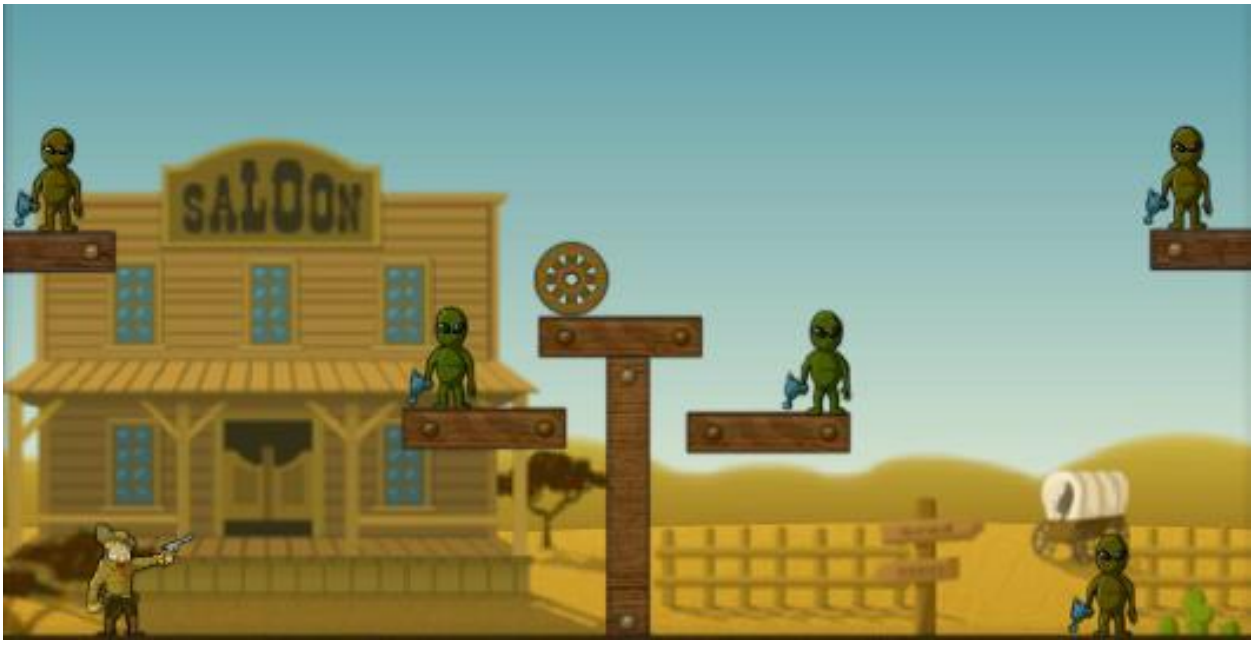
Specifikáció

Platformer játék JAVA Swing használatával

A játék egy klasszikus platformer játék, ellenségekkel és akadályokkal. A pálya teljesítéséhez le kell győzni az összes ellenséget és eljutni a pálya jobb oldalán lévő célhoz. Az ellenségek megpróbálják lövéseikkel eltalálni a játékost, a játékosnak pedig ezt legjobb tudása szerint ki kell kerülnie ha életben akar maradni.

A játékosnak meghatározott számú élete van (kevesebb élet-> nehezedik a játék), csak úgy, mint az ellenségeknek.

A játékost a WASD és Space billentyűkkel lehet irányítani. A-D-vel balra és jobbra mozogni, W-vel ugrani, S-el leguggolni, Space-el pedig lőni lehet.



Egy ötlet a játék kinézetére

A játék megvalósításához Swing-et fogok használni.

Felhasználói leírás

A menü

A menü kinézete

A menüben a New Game-re kattintva lehet új játékot indítani, a Quit pedig bezárja a játékot.

A játék végét a következő képernyő jelzi majd:

A játék

Pillanatkép a játékból

A játék során a játékosnak le kell lőnie a platformokon őrjáratozó ellenségeket. Minden ellenség négy találattal ölhető meg, a játékos 10 találatot képes túlélni. A platformokra felugrani a D+W, vagy A+W billentyűkombinációval lehet. Felfele ugrani a W egyszeri lenyomásával, balra haladni az A folyamatos lenyomásával, jobbra haladni a D folyamatos lenyomásával lehet. Lőni a bal egérgombbal való kattintással van lehetőség. A lövedék a kattintás helye felé fog haladni.

Amikor az ellenségek meglátják a játékost, lőni kezdenek, így a játékosnak át kell gondolnia a stratégiáját.

Fejlesztői leírás

Megjelenítő osztályok

A program négy olyan saját osztállyal rendelkezik, amelyeket Swing osztályból származtatok. Ez a **ScaledGifImage**, amely gif képeket képes tárolni és megjeleníteni. A **GameWindow**, ami a játékablak, képes zenét lejátszani és ebben található meg a billentyűzetet (player input) figyelő listener. Ezen a kettőn kívül még az **ImagePanel** és az **ImageLabel**, amelyek gif képek megjelenítésére alkalmas Panel és Label megvalósítások.

1. ScaledGifImage

Az egyetlen nem getter/setter függvény a **konstruktor**a. Ebben egy BufferedImage-be betöltjük a gif-et (nem kell feltétlenül gif-nek lennie, sima képeket is képes megjeleníteni). Ezután a byte adatokat átalakítjuk a defaultToolkit segítségével Image-é, majd ebből kinyerjük a megfelelő ImageIcon-t, amivel már létre tudunk hozni egy JLabel-t.

A paintComponentben elég az Image-t kirajzolni.

2. GameWindow

A **konstruktor**ban beállítom a szokásos alapvető dolgokat és felveszek egy saját kurzort, valamint létrehozom a billentyűzetet figyelő objektumot (lehet, hogy ezt célszerűbb lenne majd a startLevel()-ben megtenni).

- **public void keyHandler(KeyEvent ke)**

  A játékos inputját kezelő függvény. Az ugrások és esések kezelésére külön szálakat indít.

  Ha a karakter épp nem ugrik és van alatta talaj, akkor az A és D lenyomására balra-jobbra mozgatja.

  Ha W-t nyomtak le úgy, hogy az utolsó billentyű nem A és nem D volt, valamint a karakter épp nem ugrik, akkor egy felfelé ugrást végez.

  Ha a játékos a megadott időlimiten belül megnyomja az A vagy a D után a W-t, akkor egy ferde hajításhoz hasonló mozgást végez a karakter.

- **public void createKeyListener()**

  Ebben hozunk létre egy olyan KeyListener-t, ami a saját KeyHandler-ünket használja az inputok kezelésére, valamint váltogat az “idle” és “running” animációk között.

- **public void startLevel(Menu menu)**

  Elindít egy új szintet, valamint eltávolítja a menüt a frameről, így előtérbe helyezve a szintet.

- **public void endGame( )**

  Ha a karakter meghal, vagy meghalt az összes ellenfél, a karakter vagy a szint meghívja ezt a függvényt, ami kiírja az EndGame-t.

- **public synchronized void playAudioOnLoop(File f)**

  Az adott fájlból beolvassa a wav formátumú zenét, majd ezt loopolva lejátsza.

3. ImageLabel és ImagePanel

Az **ImageLabel és ImagePanel** csak konstruktort és setter/gettereket tartalmaz, hogy elérjem a ScaledGifImage-t amit tartalmaznak.

Karakter osztályok

Ide tartozik a **GameCharacter**, és az ebből származtatott **PlayerCharacter** és **HostileCharacter**, amik csak a gotHit függvényben és a "hp" változójukban különböznek.

1. PlayerCharacter

- **public void gotHit()**

  Ha eltalálták a játékost csökkenti eggyel az életét. ha a játékos meghalt, meghívja a jelenlegi játékablak endGame( ) függvényét, és vége a játéknek.

2. HostileCharacter

- **public void gotHit()**

  Ha eltalálták az ellenséges karaktert, akkor csökkenti eggyel az életét. Ha meghalt, meghívja a szint számláló függvényét, amely ha az összes ellenség meghalt, meghívja az ablak endGame() függvényét.

3. GameCharacter

A **GameCharacter** csak konstruktort, valamint gettereket és settereket tartalmaz

Játék kellék osztályok.

Ide tartoznak a **Bullet, Level, GameTile** és **Menu** osztályok.

1. Bullet

A Bullet osztály a **konstruktorában** vagy a **public void setUpBullet()**(amit egy setDestination után értelmes csak meghívni) segítségével kiszámolja a pályáját, ezen kívül csak getter és setter függvények találhatók.

2. Level

A Level osztály létrehozza a demó szintjét. Kezeli a lövéseket a klikkelés és az ellenségek lövés algoritmusa segítségével. Ha kilő egy golyót akkor keres a maradék 29 körül egy új szabadot, és a következő kilőtt már az lesz.

- **public void shootBullet(Point p, GameCharacter hostile) és public void ShootBullet(MouseEvent me)**

  A Point p és GameCharacter hostile átadott értékekkel meghívott függvény az ellenségek által használandó, míg a MouseEvent me értékkel meghívott függvény a játékos klikkelése során hívódik meg. Mindkét függvény ellenőrzi, hogy elég sok idő telt-e el az utolsó lövés óta, majd beállítja, hogy ki lőtte a lövedéket (ez azért fontos, hogy ne lőjék magukat "lábon" a karakterek). Ezután elhelyezi a karakterhez és elindítja a megfelelő szálat.

- **public boolean checkHit(Point p)**
  Ellenőrzi, hogy a karakter helyzete által meghatározott Rectangle ütközik-e bármelyik platformmal, vagy az alappal.
- **public boolean checkBulletHit(Point p, GameCharacter character)**

  Ellenőrzi, hogy az adott pont egy tile, alap, vagy karakter része-e. Az átadott karakter kimarad a vizsgálatból, kiküszöbölve a "lábonlövést".

- **public void searchNewFreeBullet()**

  Lineáris kereséssel új szabad golyót keres (egy golyó szabad, ha (0,0)-n van).

- **public void deadCounter()**

  Ha egy ellenség meghal, ezt a függvényt hívja meg. Ha az összes ellenség meghalt, meghívja az endGame() függvényt.

3. GameTile

Ez az osztály megkülönbözteti a platformokat az alaptól.

4. Menu

Ez a menü osztálya. Tartalmazza a hátteret és a klikkelhető területek képeit. Ha lenyomják a "New Game"-et, akkor meghívja az ablak startLevel függvényét. Ha a Quit-re kattintanak egy System.exit(0) hívást végez.

Thread osztályok

Ide tartoznak a **BulletThread, FallingThread,JumpingThread,UpwardsJumpThread** osztályok.

1. BulletThread

Tárolja a kilövendő golyót. Futás során 20-asával változtatja a golyó helyzetét, újrafesti. Ha eltalál valamit, akkor kilép a végtelen ciklusból és (0,0) helyzetbe helyezi a golyót.

2. FallingThread

Tárolja az esésben lévő karaktert. A ciklust addig fut, ameddig nem talál el valamit és akkor hívódik meg a szál, ha éppen nincs semmi a karakter alatt. Esés közben ha a karakternek volt eredetileg X irányú sebessége ferde hajítás szerű mozgást végez. Amikor a karakter földet ér megállítja.

3. JumpingThread

Ferde hajítás szerű ugrást végrehajtó szál. Ha beleütközik a karakter valamibe, akkor egyenesen leesik. A szál végén egy mesterséges “A” vagy “D” billentyűlenyomást ad le, hogy a W és A vagy D folyamatos lenyomása mellett lehessen folyamatosan ugrálni. Ezt sajnos a sima balra/jobbra menéssel nem tudtam megvalósítani.

4. UpwardsJumpThread

Egyenesen felfele ugrást megvalósító szál. Ha ugrás közben beveri a fejét valamibe, akkor onnan kezd el esni 0 sebességről.

Összefoglalva az osztályok közti kommunikációt és a program menetét:

A játék elején létrejön a GameWindow amihez hozzáadódik a Menu ImagePanel. Ez az objektum figyeli a kattintásokat, és ha a “New Game”-re kattintanak, akkor meghívja az ablak szintet előkészítő függvényét, amely egyben eltávolítja a menüt az ablakból. A szint figyeli azt, hogy hova klikkel a játékos, az ablak figyeli azt, hogy milyen billentyűt nyom le. A HostileCharacterek mozgás közben figyelik a játékos helyzetét, ha egy Y szintre kerül velük akkor lőnek a BulletThread és a Level segítségvel. Ha a játékos ugrik vagy esik az ablak elindít egy megfelelő ugrással/eséssel kapcsolatos szálat. Ha egy karakter lövést kap levon a saját életéből, ha eközben meghalt értesíti erről a szintet (ha ellenség), vagy az ablakot (ha játékos). Ha az összes ellenség meghalt a szinten, akkor a szint értesíti az ablakot erről. Mindkét esetben az ablak meghívja a saját, játékot befejező függvényét.