Programozási Technológia III. Beadandó
Készítette: Halmos Titanilla B91V1Z

Feladat leírása

Maci Laci (Yogi Bear)

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 menübő.

Feladat elemzése

A feladat azt várja tőlünk, hogy egy felső nézetből irányítható játékot készítsünk. Egy mozgásra képes fő karaktert, magától mozgást folytató őröket, akadályokat, és összegyűjthető elemeket kell létrehozni.

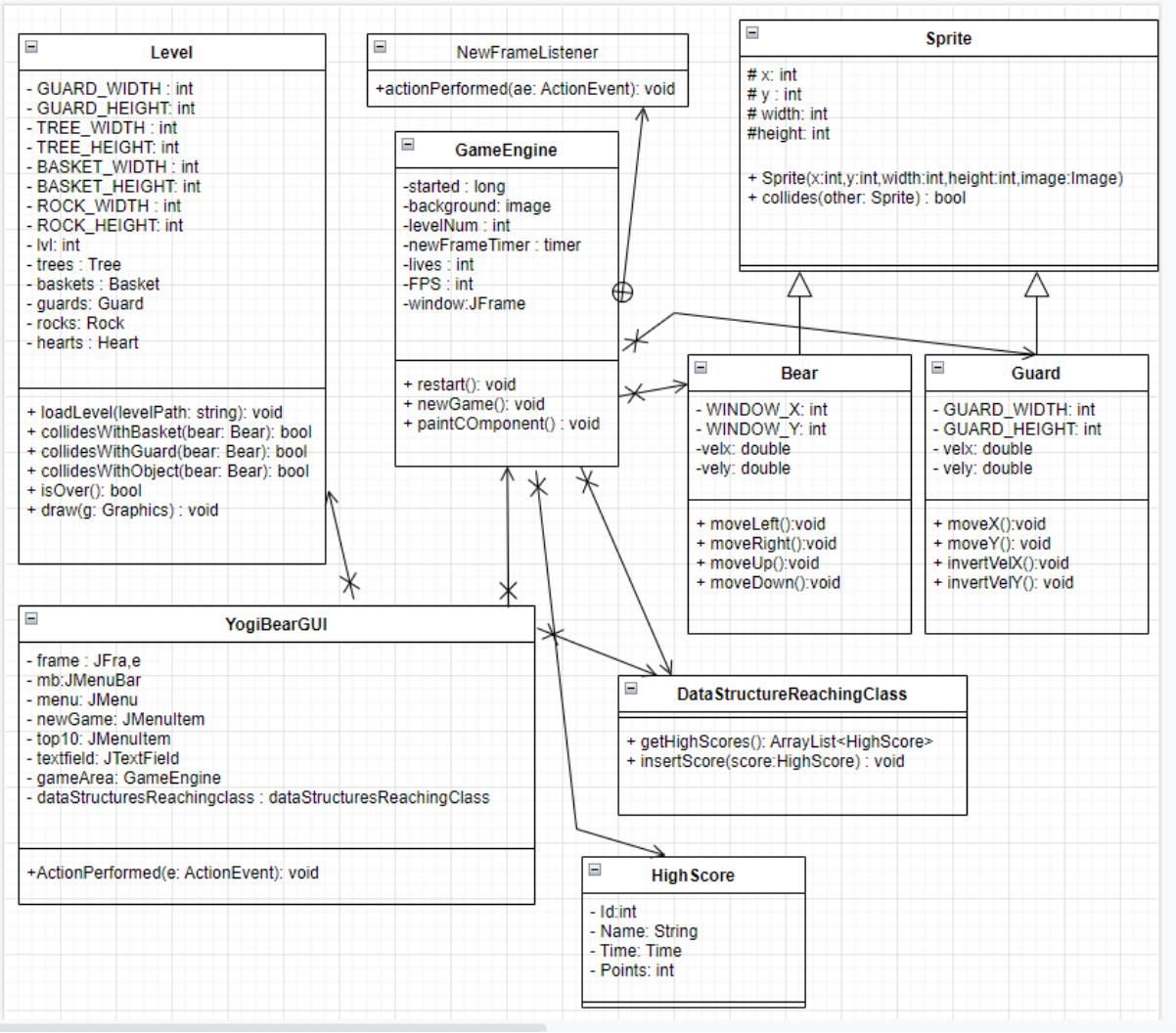
A fő karaktert a billentyűzeten meghívott eseményfigyelővel lehet mozgatni.

Az őr / őrök vízszintes / függőleges mozgást végeznek, szintén egy eseményfigyelő segítségével.

Az összegyűjthető elemeket ha elérí a fő karakter akit mozgatunk, akkor az lekerül az ablakról.

Az akadályok és a fő karakter találkozásakor a fő karakter nem tud mozgást folytatni az akadályok irányába.

Osztálydiagram



Implementáció

GameEngine: A játék fő mozgató eleme, itt rendeljük hozzá a fő karakterhez (bear) az KeyStroke figyelőt, ami a mozgatást végzi.

```
public GameEngine(JFrame window) {
        super();
        try {
                this.dataStructuresReachingClass = new
                                DataStructuresReachingClass();
                this.highScores =
                                dataStructuresReachingClass.getHighScores();
        }
        catch (Exception ex) {
                System.out.println(ex.getMessage());
```

```
        }
        this.window = window;
        this.menuBar = window.getJMenuBar();
        this.timeLabel = new JLabel();
        Time time = new Time(0);
        time.setHours(0);
        timeLabel.setText(time.toString());
        menuBar.add(timeLabel);
        lives = 3;
        background = new ImageIcon("src/park.jpg").getImage();
        Image bearimage = new ImageIcon("src/YogiBear.png").getImage();
        bear = new Bear(5,5,60,60,bearimage);
        Image guardimage= new ImageIcon("src/guard.png").getImage();
        Image heartimage = new ImageIcon("src/heartfinal.png").getImage();
        heart = new Heart(720,10,20,20,heartimage);
        heart2 = new Heart(740,10,20,20,heartimage);
        heart3 = new Heart(760,10,20,20,heartimage);
        hearts = new ArrayList<>();
        hearts.add(heart);
        hearts.add(heart2);
        hearts.add(heart3);
        guard = new Guard(300,300,60,60,guardimage);
        this.getInputMap().put(KeyStroke.getKeyStroke("RIGHT"), "move
                        right");
        this.getActionMap().put("move right", new AbstractAction() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        int x = bear.getX();
                        int y = bear.getY();
                        bear.moveRight();
                        if(level.collidedWithObject(bear)) {
                                bear.setX(x);
                                bear.setY(y);
                        }
                        window.repaint();
                }
        });
        this.getInputMap().put(KeyStroke.getKeyStroke("DOWN"), "move
                        down");
        this.getActionMap().put("move down", new AbstractAction() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        int x = bear.getX();
                        int y = bear.getY();
                        bear.moveDown();
                        if(level.collidedWithObject(bear)) {
                                bear.setX(x);
                                bear.setY(y);
                        }
                        window.repaint();
                }
        });
        this.getInputMap().put(KeyStroke.getKeyStroke("LEFT"), "move
                        left");
        this.getActionMap().put("move left", new AbstractAction() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        int x = bear.getX();
                        int y = bear.getY();
                        bear.moveLeft();
```

```
                        if(level.collidedWithObject(bear)) {
                                bear.setX(x);
                                bear.setY(y);
                        }
                        window.repaint();
                }
        });
        this.getInputMap().put(KeyStroke.getKeyStroke("UP"), "move up");
        this.getActionMap().put("move up", new AbstractAction() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        int x = bear.getX();
                        int y = bear.getY();
                        bear.moveUp();
                        if(level.collidedWithObject(bear)) {
                                bear.setX(x);
                                bear.setY(y);
                        }
                        window.repaint();
                }
        });
        restart();
        newFrameTimer = new Timer(1000 / FPS, new NewFrameListener());
        newFrameTimer.start();
        this.started = System.currentTimeMillis();
}
```

Az fő karakter az összegyűjtendő elemmel való “találkozását” megvalósító függvény:

```
public boolean collidesWithBasket(Bear bear) {
    Basket collidedWithBasket = null;
    for (Basket basket : baskets) {
        if (bear.collides(basket)) {
            collidedWithBasket = basket;
            break;
        }
    }
    if (collidedWithBasket != null) {
        baskets.remove(collidedWithBasket);
        return true;
    }
    return false;
}
```

Esemény – Eseményvezérlő párok

YogiBear – KeyStroke map: figyeli a billentyűzet eseményeit, és ha egyezést talál meghívja YogiBear az iránynak megfelelő move függvényét

Menü New Game gombja – actionPerformed: a gomb lenyomásakor a GameEngine newGame() függvénye meghívódik és új játékot kezd

Menü top 10 játékos gombja – actionPerformed: a gomb lenyomásakor a dataStructureReachingClass-ból elért tömböt kiírja egy új ablakban

Frame - NewFrameListener osztály eseményvezérlője: 1000 / FPS másodpercenként meghívódik játék közben: ez mozgatja az őrt / őröket, illetve frissíti a képet, például ha YogiBear megszerzett egy kosarat.