

Quadterm 2

(streda 13.5. !9:20! – 11:30)

Bude:

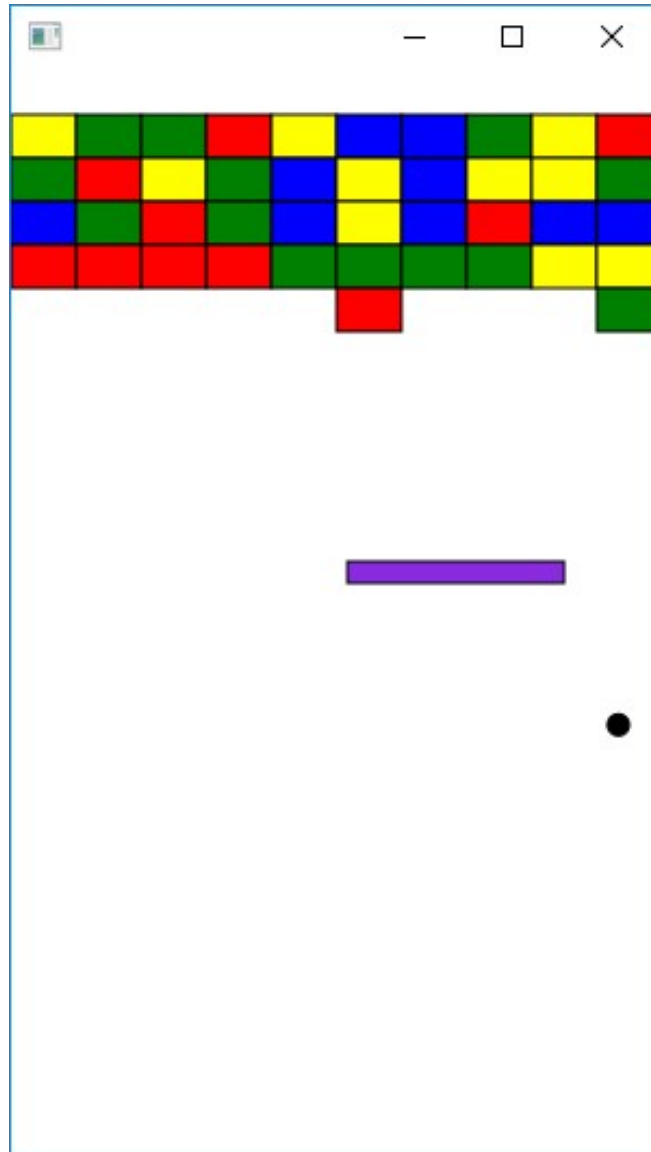
- simulácia niečoho, čo sa hýbe
 - Thread/Timeline/AnimationTimer
- odchyťavanie udalostí
myš/klávesnica
- kreslenie do
 - Pane
 - `getChildren().clear()`
 - `new Rect(...)`
 - `new Circle(...)`
 - `new ImageView()`
 - `getChildren().add(...)`
 - Canvas
 - `gc = getGraphicContext`
 - `gc.strokeLine`
 - `gc.fillRect()`
 - `gc.drawImage()`

Nebude:

- serializácia
- zložitejší layout
- import project –
asi nebude template 😞
- unit testy 😊

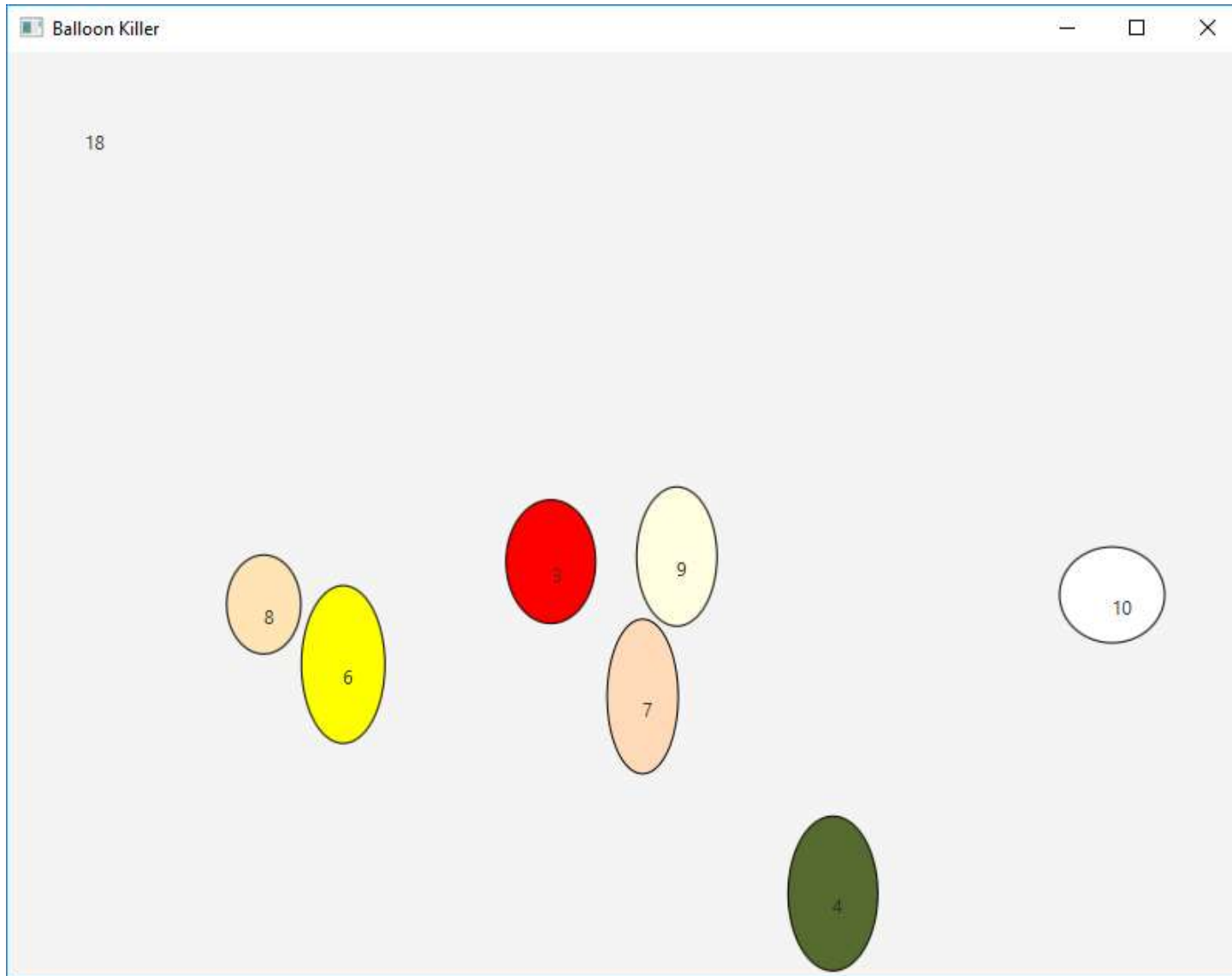
Arkanoid

Quad2 2015



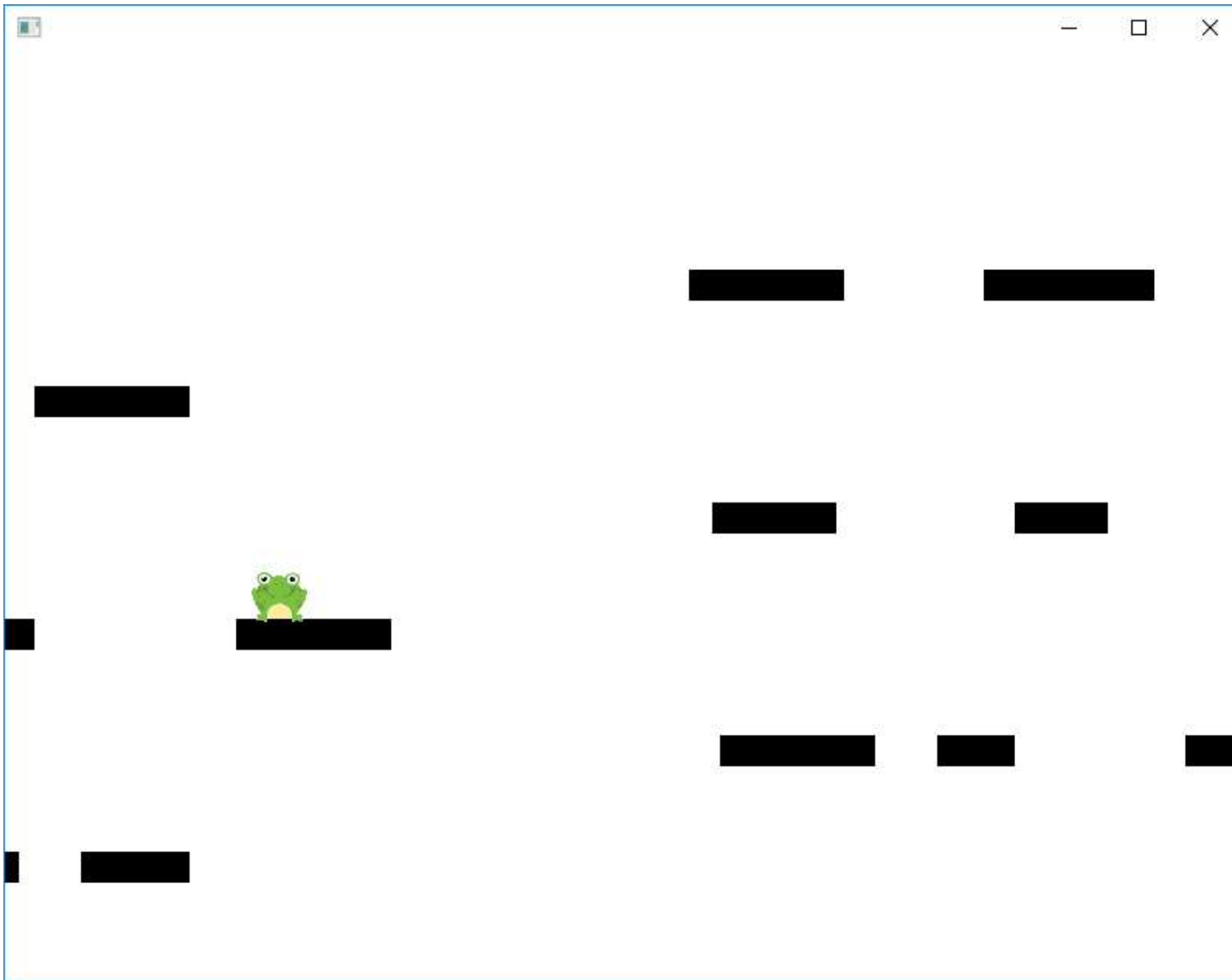
Balloon Killer

Quad2 2016



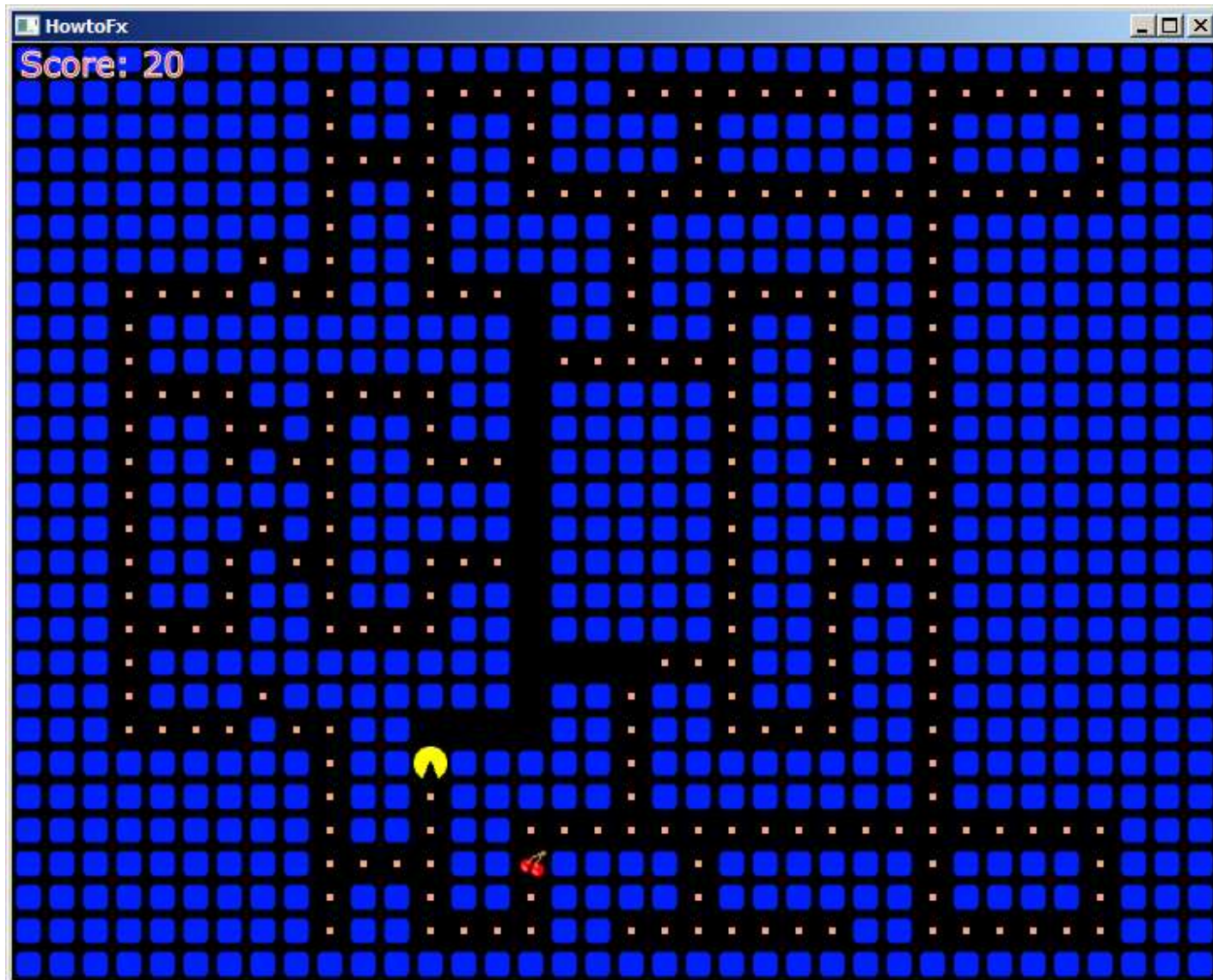
Žabky

Quad2 2017



Pacman

Quad2 2018



Tanky

Quad2 2019

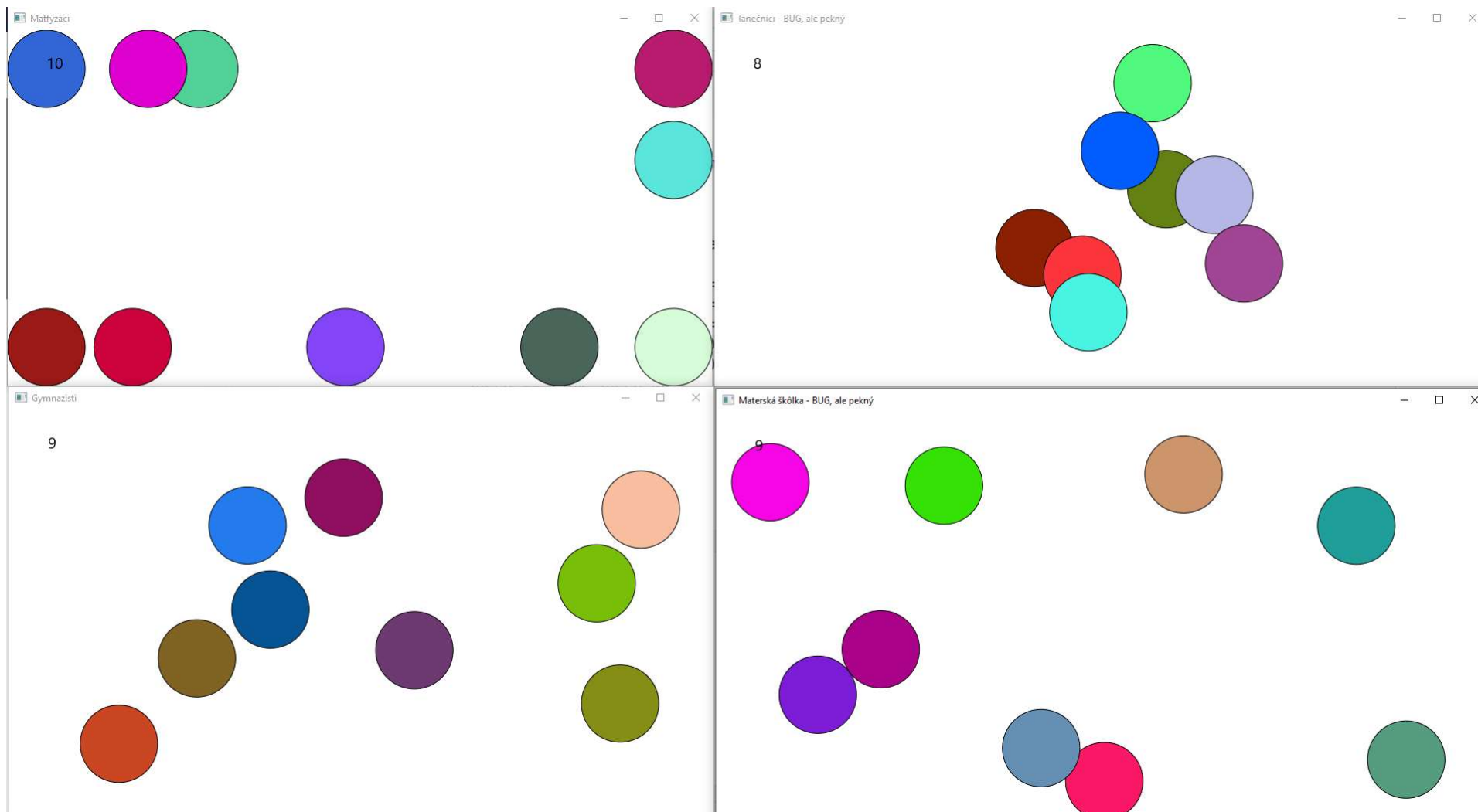


Cenzorovaná verzia



Späť do školy

Quad2 2020



JavaFx

pokračovanie



Už vieme (treba pre quadterm2):

- kresliť do Canvas, vložiť Canvas->Pane->Scene->Stage, [HowToWithFx](#)
- simulovať dej pomocou Thread/Timeline/AnimationTimer,
- prekreslovať GUI komponenty pomocou Platform.runLater
- odchytiť udalosti od(ActionEvent/KeyEvent/MouseEvent,
- aj to že uhol dopadu sa rovná uhlu odrazu ☺
- rôzne spôsoby návrhu jednoduchšej (pravouhlej) hry (Grid/Canvas/Button)

Dnes bude:

- aspekt škálovateľnosti (re-zoom hracej plochy),
- perzistencia (ukladanie dát),
- 3D JavaFX od Lukáša G.,

Zdroj a literatúra:

- [Introduction to Java Programming, !!!!Tenth Edition](#)

Cvičenia: jednoduché aplikácie s GUI:

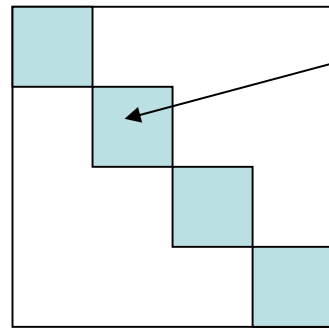
- škálovateľná logická hra
- 3D Labyrint

Hracia plocha

hracia plocha je často šachovnica rôznych rozmerov. Ako ju implementujeme:

1. jeden veľký canvas v Pane-li:

- musíme riešiť transformáciu pixelových súradníc do súradníc hracej plochy:

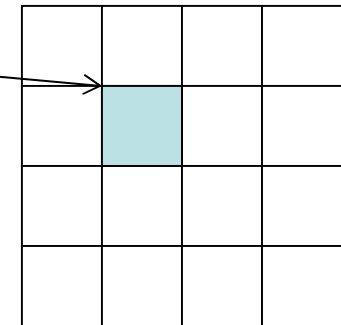


`[event.getX(),event.getY()]->[1,1]`

- a naopak, v metóde `paintMôjCanvas/paintMôjComponent [i,j] -> [pixelX, pixelY]`

2. grid canvasov/Pane-lov:

- každý canvas/panel má svoje súradnice od `[0,0]`
- každý canvas/panel má svoj mouse event handler
- každý canvas panel má svoju metódu `paint/paintMôjCanvas`
- veľkosť gridu upravíme podľa veľkosti obrázkov,
resp. veľkosť obrázku upravíme podľa veľkosti panelu



3. grid buttonov/Button-ov, Button môže mať obrázok ako ikonu

Pexeso

bolo na cvičení

```
public class Pexeso extends Application {
    State state = new State();
    Playground playground;
    public class Playground extends GridPane {
        public class Cart extends Pane { ... }
    }
}

// POZOR, TOTO NEMOŽE BYŤ VNORENÁ TRIEDA, lebo ... ani public
class State implements Serializable {
    private static final long serialVersionUID = 918972645L;
    public class CartObject implements Serializable {
        private static final long serialVersionUID = 911775039L;
        int id;
        boolean visible = false;
        transient ImageView pikaImage; // neserializovať
    }
}
```

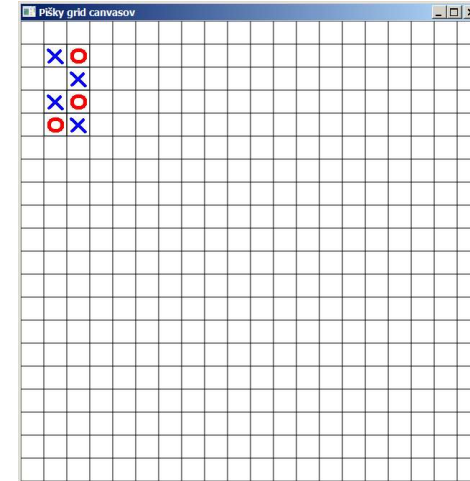
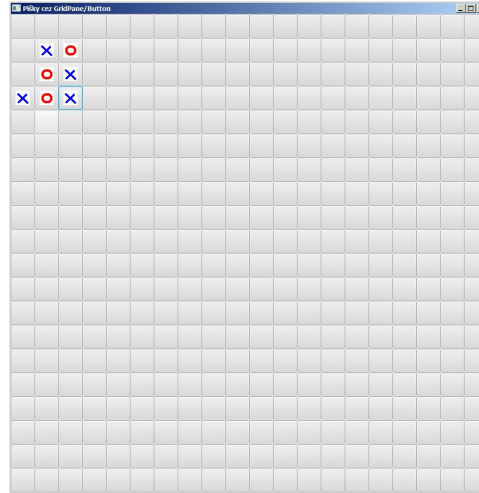
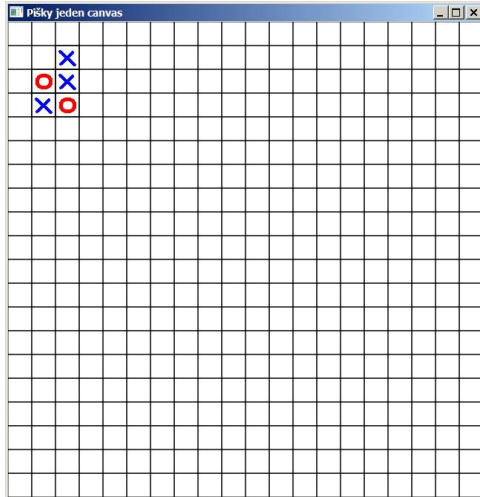
- **transient** znamená, že nechceme serializovať,
- väčšina JavaFX objektov nie je serializovateľná !!! a dostanete **NotSerializableException**...
- **transient static** a **transient final** nerobí nič

Súbor: [Pexeso.java](#)

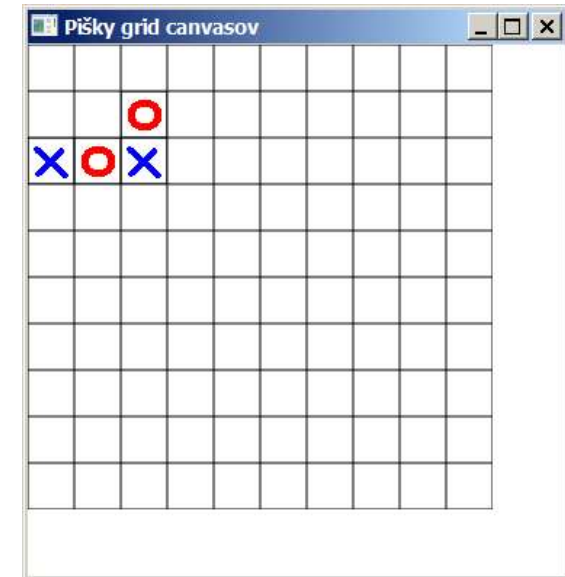
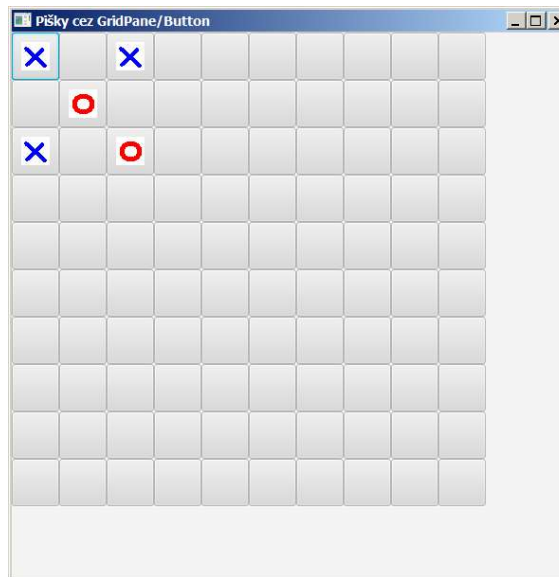
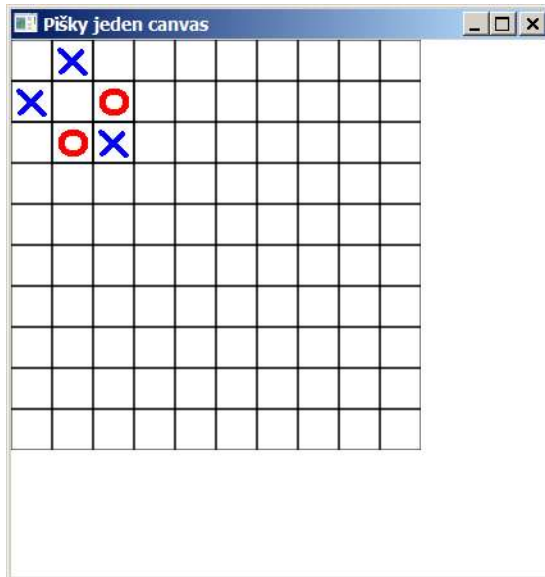


Škálovateľnosť

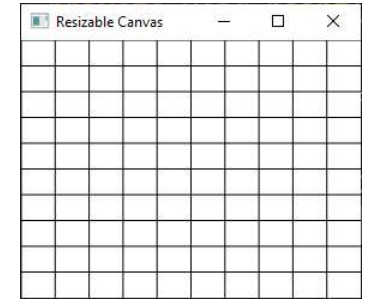
- Škálovateľnosť hry (miesto 10x10 chceme hrať 20x20):



- Škálovateľnosť GUI (zmeníme rozmer okna):



Škálovateľný Canvas



```
final int SIZE = 10;

class Playground extends Canvas {
    public Playground() { // ak sa zmení veľkosť, prekreslí celý canvas
        widthProperty().addListener(event -> paint());
        heightProperty().addListener(event -> paint());
    }

    private void paint() {
        double width = getWidth(); // zisti aktuálnu veľkosť, šírku
        double height = getHeight(); // a výšku
        GraphicsContext gc = getGraphicsContext2D(); // kreslí pravouhlu mriežku
        gc.clearRect(0, 0, width, height); // ale najprv si to vygumuj
        gc.setStroke(Color.BLACK);
        for(int i = 0; i<SIZE; i++) gc.strokeLine(0, i*height/SIZE, width, i*height/SIZE);
        for(int i = 0; i<SIZE; i++) gc.strokeLine(i*width/SIZE, 0, i*width/SIZE, height);
    }
}

public void start(Stage stage) throws Exception {
    Playground pg= new Playground();
    Pane p = new Pane(pg);
    pg.widthProperty().bind(p.widthProperty()); // pg.width = p.width
    pg.heightProperty().bind(p.heightProperty()); //pg.height=p.height
    stage.setScene(new Scene(p, 400, 400));
}
```

Súbor: [ResizableCanvas.java](#)

Properties & Bindings

```
DoubleProperty polomer = new SimpleDoubleProperty();
DoubleProperty priemer = new SimpleDoubleProperty();
priemer.bind(polomer.multiply(2)); // priemer = 2*polomer

DoubleProperty obvod = new SimpleDoubleProperty();
obvod.bind(polomer.multiply(2).multiply(Math.PI)); // obvod = 2*PI*polomer

NumberBinding stvorec = Bindings.multiply(polomer, polomer);
DoubleProperty obsah = new SimpleDoubleProperty(); // stvorec=polomer*polomer
obsah.bind(stvorec.multiply(Math.PI)); // obsah = PI*stvorec

// polomer.bind(priemer.divide(2)); // cyklická referencia, to nedá ☹️
for (double r = 0; r < 2; r += 0.5) {
    polomer.set(r);
    // obvod.set(r); // génus nie je!
    System.out.printf(
        "polomer=%6.2f, priemer=%6.2f, obvod=%6.2f, obsah=%6.2f\n",
        polomer.getValue(),
        priemer.getValue(), obvod.getValue(), obsah.getValue());
}
```

| | | | | | | | |
|----------|-------|----------|-------|--------|-------|--------|------|
| polomer= | 0,00, | priemer= | 0,00, | obvod= | 0,00, | obsah= | 0,00 |
| polomer= | 0,50, | priemer= | 1,00, | obvod= | 3,14, | obsah= | 0,79 |
| polomer= | 1,00, | priemer= | 2,00, | obvod= | 6,28, | obsah= | 3,14 |
| polomer= | 1,50, | priemer= | 3,00, | obvod= | 9,42, | obsah= | 7,07 |

1. Riešenie škálovateľné jeden Canvas

```
Piskyground pg = new Piskyground(); // one big Canvas
Scene scene = new Scene(new Group(pg), 500, 500); // najaká iniciálna veľkosť
pg.widthProperty().bind(scene.widthProperty()); // pg.width = scene.width
pg.heightProperty().bind(scene.heightProperty()); // pg.height = scene.height
pg.paintAll(); // inak by sa nič nevykreslilo

scene.widthProperty().addListener(event -> pg.paintAll()); // changeListener
scene.heightProperty().addListener(new ChangeListener<Number>() { //full verzia
    @Override
    public void changed(ObservableValue<? extends Number> observableValue,
        Number oldSceneHeight, Number newSceneHeight) {
        System.out.println("Height: " + newSceneHeight);
        pg.paintAll();
    }
});

primaryStage.setTitle("Resizable Pišky jeden canvas");
...
```

Transformácie



```
class Piskyground extends Canvas {                                // sú komplikovanejšie
    // a už záleží na x,y lebo plocha môže byť obĺžnik

    private double cellWidth()      { return getWidth()/SIZE; }
    private double cellHeight()     { return getHeight()/SIZE; }
    private int   getRow(double pixelY) { return (int) (pixelY / cellHeight()); }
    private int   getCol(double pixelX) { return (int) (pixelX / cellWidth()); }
    private double getPixelX(int row)  { return row * cellHeight(); }
    private double getPixelY(int col)  { return col * cellWidth(); }
} ! Napriek tomu, že ide o lineárne transformácie, abstrahujte ich do metód !

public void paintCell(int i, int j) {
    Image imageO = // keď potrebujem obrázok danej šírky a výšky
        new Image("o.gif", cellWidth()-2, cellHeight()-2, false, false);
    Image imageX =
        new Image("x.gif", cellWidth()-2, cellHeight()-2, false, false);
    . . . .
```


Hra 15

BoundsProperty listener

```
public class Hra15 extends Application {  
    final int SIZE = 4;    final int COLS = SIZE;    final int ROWS = SIZE;  
    @Override  
    public void start(final Stage primaryStage) throws Exception {  
        GridPane gp = new GridPane();  
        for (int i = 0; i < 16; i++) {           // vytvorí hraciu plochu  
            Button button = (i == 15) ? new Button("") : new Button("" + (i + 1));  
            gp.add(button, i % COLS, i / COLS); // mod, div=súradnice políčka i  
        }  
        gp.layoutBoundsProperty().addListener( // ak sa zmenia rozmery gp  
            (observable, oldBounds, newBounds) -> {  
                double cellHeight = newBounds.getHeight() / ROWS;  
                double cellWidth = newBounds.getWidth() / COLS;  
                for (final Node child : gp.getChildren()) {  
                    final Control tile = (Control) child;  
                    tile.setPrefSize(cellWidth, cellHeight);  
                } // prekreslí všetky Node v gp  
            });  
    }  
}
```



Súbor: [Hra15.java](#)

2. Riešenie škálovateľné

fitWidth/HeightProperty

[Súbor: PiskvorkyGridButtonResizable.java](#)

@Override

```
public void start(Stage primaryStage) {
    Piskyground pg = new Piskyground();
    pg.layoutBoundsProperty().addListener((observable, old, newBounds) -> {
        for (final Node child : pg.getChildren()) { // ak sa zmení rozmer pg
            final Control tile = (Control) child; // zmeň veľkosti buniek
            tile.setPrefSize(newBounds.getWidth() / SIZE,
                             newBounds.getHeight() / SIZE);
        }
    });
}

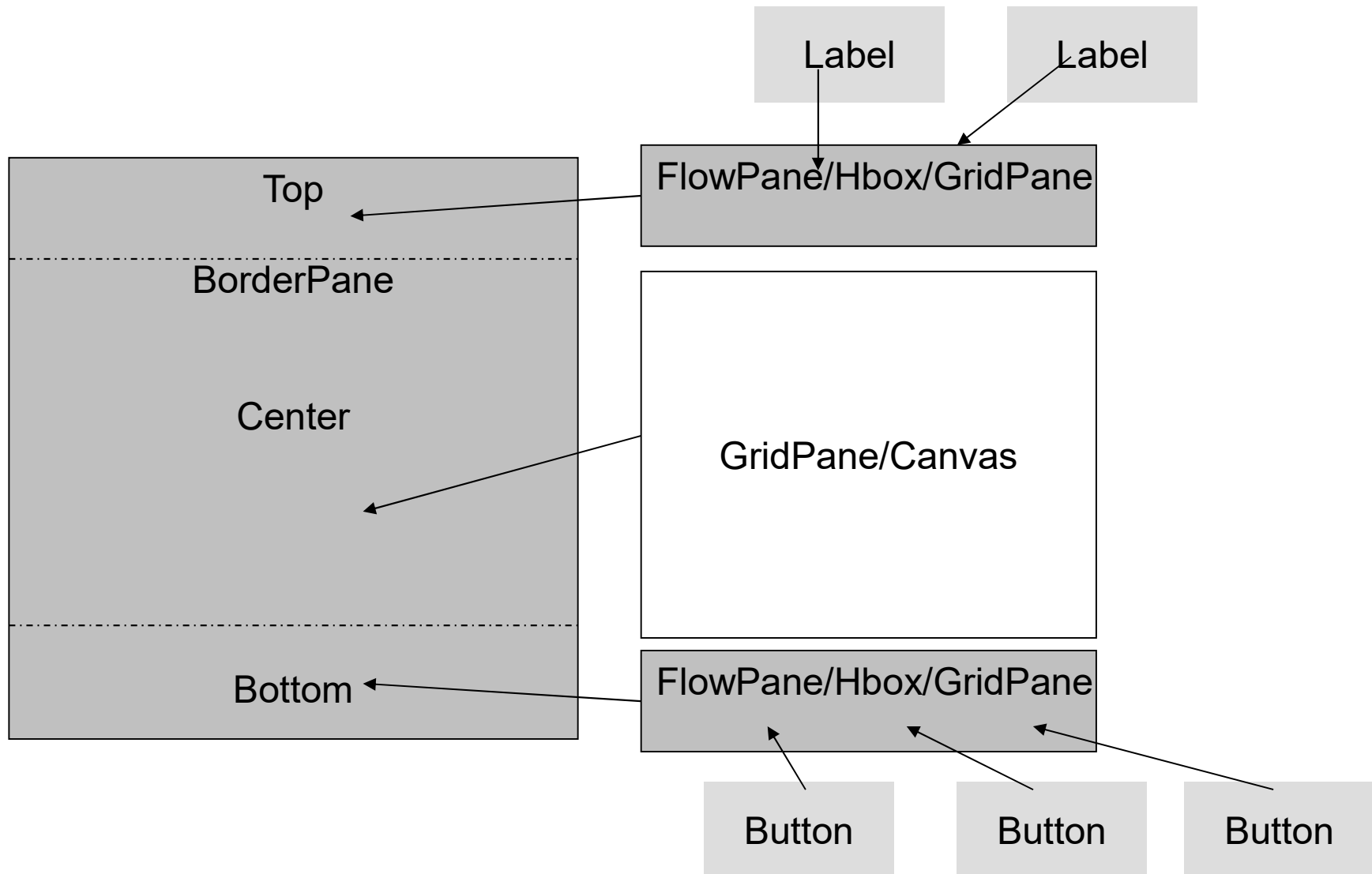
class PiskyCell extends Button {
    ImageView imageO = new ImageView(new Image("o.gif"));
    ImageView imageX = new ImageView(new Image("x.gif"));
    public PiskyCell(int i, int j) {
        setMinSize(50, 50); // menej nedovolí
        imageX.fitWidthProperty().bind(widthProperty()); // X.width = this.width
        imageX.fitHeightProperty().bind(heightProperty()); // X.height = this.height
        imageO.fitWidthProperty().bind(widthProperty().subtract(4)); // 2px okraj
        imageO.fitHeightProperty().bind(heightProperty().subtract(4)); // 2px lem
    }
}
```

3. Riešenie škálovateľné

Súbor: [PiskvorkyCanvasResizable.java](#)

```
public class PiskvorkyGridCanvasResizable extends Application {
    pg = new Piskyground();
    scene.widthProperty().addListener((observableValue, old, newSceneWidth)->{
        pg.prefWidth((double) newSceneWidth);
        pg.paint();
    });          // to isté pre height
class Piskyground extends GridPane {
    public Piskyground() {
        for (int i = 0; i < SIZE; i++) for (int j = 0; j < SIZE; j++) {
            PiskyCell pc = canvasGrid[i][j] = new PiskyCell(i, j);
            add(pc, j, i);
            pc.widthProperty().bind(widthProperty().divide(SIZE)); // tiež height
        }
    }
class PiskyCell extends Canvas {
    public void paintCell() {
        GraphicsContext gc = getGraphicsContext2D();
        Image imageX=new Image("x.gif",getWidth()-2,getHeight()-2,false,false);
        Image imageO=new Image("o.gif",getWidth()-2,getHeight()-2,false,false);
```

Scéna hry



Layout

```
Piskyground pg = new Piskyground();           // pôvodná hracia plocha
BorderPane bp = new BorderPane();               // vonkajší rámec
bp.setCenter(pg);

HBox labelPane = new HBox(                      // vrchný panel, FlowPane, GridPane, ...
    new Label("Score:"), lbScore = new Label("0"),
    new Label("Elapsed time:"), lbTime = new Label("0"),
    new Label("Next:"), lbOnMove = new Label("o"));
labelPane.setSpacing(20);                       // hrubý layout, s tým sa dá vyhrať...
lbScore.setFont(Font.font(18)); ...
bp.setTop(labelPane);                           // umiestnime na vrch

HBox buttonPane = new HBox(                     // spodný panel plný tlačidiel, gombíkov
    btnLoad = new Button("Load"), btnSave = new Button("Save"),
    btnQuit = new Button("Quit"));
buttonPane.setSpacing(50);
bp.setBottom(buttonPane);                       // umiestnime na spodok
```

Control

```
btnQuit.setOnAction(event -> Platform.exit());
```

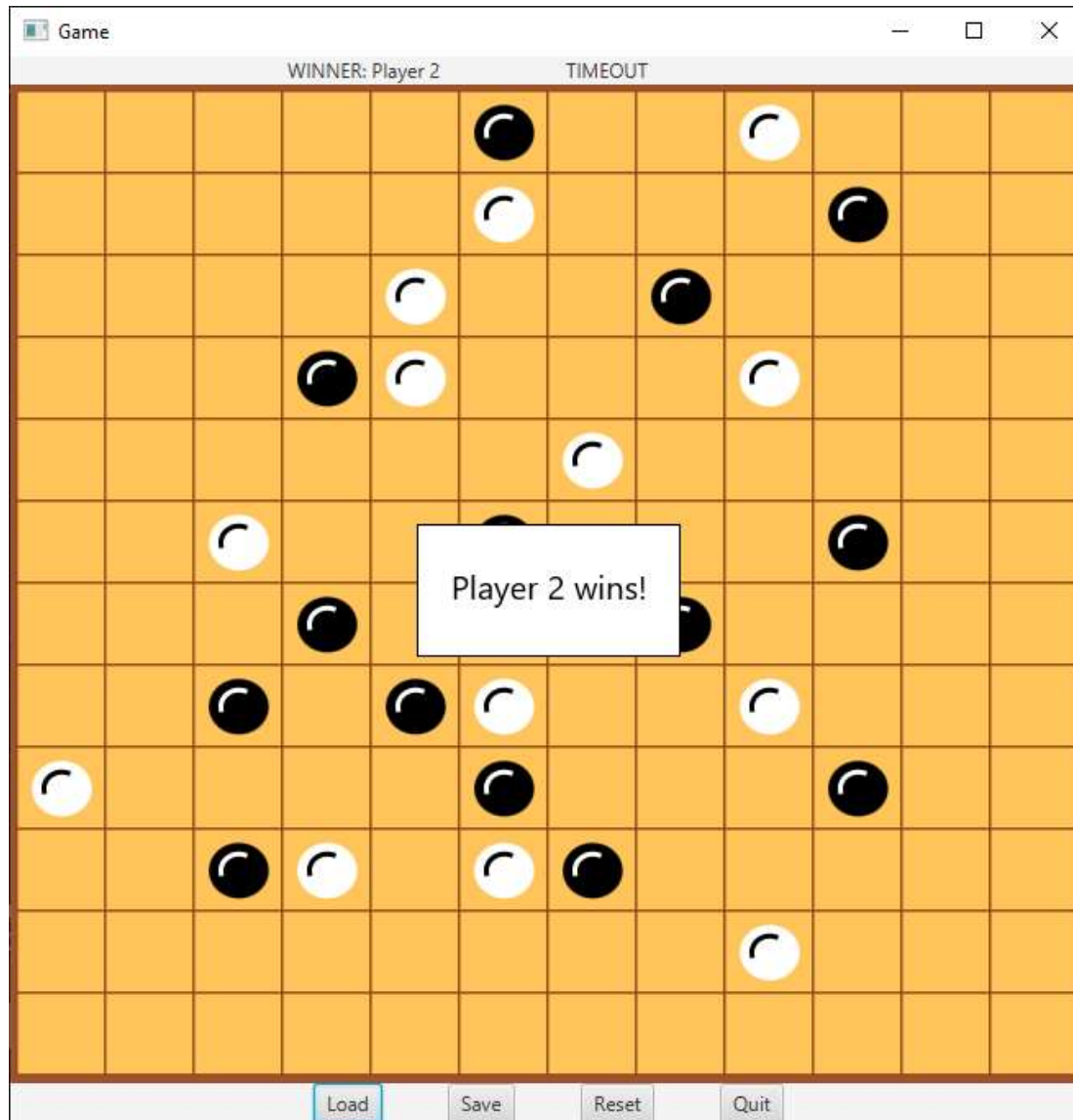
```
btnLoad.setOnAction(event -> {                                // načítanie konfigurácie
    try {
        ObjectInputStream is=new ObjectInputStream(new FileInputStream("p.cfg"));
        ps = (PiskyState) is.readObject();
        is.close();
        pg.paintAll();                                         // prekresli scénu, inak sa zmení len stav
    } catch (Exception e) { e.printStackTrace();}
} );
```

```
btnSave.setOnAction(event -> {                                // uloženie konfigurácie
    try {
        ObjectOutputStream fs=new ObjectOutputStream(new FileOutputStream("p.cfg"));
        fs.writeObject(ps);
        fs.close();
    } catch (Exception e) { e.printStackTrace(); }
} ); // pozor ! Väčšina javafx objektov nie je serializovateľná, ani Image...
```

Timer

```
Timeline tl = new Timeline(1000);           // počítame spotrebovaný čas
tl.setCycleCount(Timeline.INDEFINITE);
tl.getKeyFrames().add(new KeyFrame(Duration.seconds(1), event -> {
    ps.elapsedTime++;
    Platform.runLater(new Runnable() {
        @Override
        public void run() {
            lbTime.setText(""+ps.elapsedTime); // a prekreslujeme do info políčka
        }
    });
}));
tl.play();
```

Scéna hry



Pokračovanie

- Lukáš Gajdošech: JavaFX3D
- <https://www.youtube.com/playlist?list=PLUtV5iyaCT5GKtStZiVfGb6JbN0cmQ0gJ>

Zdrojáky:

- https://drive.google.com/file/d/1KNwE47_6qlugosRKQ-0priXy-t0-32Yc/view

Zajtra cvičenie:

- JavaFX3D (labyrint) alternatíva FX2D (škálovateľná pravouhla hra)

Posledná DÚ-C

- JavaFX3D (labyrint) alternatíva FX2D (škálovateľná pravouhla hra)