README

Feladatok:

- Rajzolás a raw image-re
- Kimenteni a körbe rajzolt képet
- Kimenteni a maszkot
- Mozgatásnál meglegyen az eddigi körbe rajzolás
    - Külön canvas kimentés az eredeti képek mellé

      majd annak betöltése a képekkel együtt?

    - Mivel négy kép van ezért a konkrét négyes párokhoz kimenteni?
    - Mozgatásnál meglegyen az eddigi körbe rajzolás
- Kezelni a zoomolást és mozgatást?
    - Esetleg tilteni a zoomolva rajzolást és csak

      nagyítani a canvasta a képpel?
- Színek: két szín ami váltja hogy positive vagy negatív FLAG-el egyen e kimentve a kép

Eszközök:

- Raw image elhelyezkedés:

MainPageControlleren belül a **makeGridViewer** metódus:

```
447     private static void makeGridViewer() {
448         Stage gridViewer = new Stage();
449         gridViewer.initStyle(StageStyle.TRANSPARENT);
450         gridViewer.initModality(Modality.NONE);
451         gridViewer.initOwner(pStage);
452 
453         StackPane stackPane = new StackPane();
454         stackPane.getChildren().add(grid);
455         stackPane.getStyleClass().add("Custom-BorderOnly");
456         stackPane.setOnScroll(event -> {...});
474         stackPane.setOnMousePressed(event -> {...});
482         stackPane.setOnMouseDragged(event -> {...});
493         stackPane.setOnKeyReleased(MainPageController::handleKeyPressed);
494 
495 
496         Scene scene = new Scene(stackPane, width: 1000, height: 1000);
497         scene.addEventFilter(KeyEvent.KEY_RELEASED, MainPageController::handleKeyPressed);
498         gridViewer.setX(300);
499         gridViewer.setY(70);
500         gridViewer.setScene(scene);
501         gridViewer.sizeToScene();
502         gridViewer.show();
503     }
```

Készít egy új stage-et (ez nem lényeg) a **stackPane** tartalmazza

a gridPane-t amiben a képek vannak tehát arra lehet építeni.

Nyugodtan lehet globálissá tenni hogy el tudd érni.

- Mozgatás:

MainPageControlleren belül a **handleKeyPressed** metódus:

```
531     static void handleKeyPressed(KeyEvent keyEvent){
532         resetMove();
533         if (isMoveReady) {
534             if (keyEvent.getCode() == KeyCode.UP){
535                 PictureRenderer.inputHandler(INPUT_UP, qualityMap.get(qualityLvl));
536             }
537             else if (keyEvent.getCode() == KeyCode.DOWN){
538                 PictureRenderer.inputHandler(INPUT_DOWN, qualityMap.get(qualityLvl));
539             }
540             else if (keyEvent.getCode() == KeyCode.LEFT){
541                 PictureRenderer.inputHandler(INPUT_LEFT, qualityMap.get(qualityLvl));
542             }
543             else if (keyEvent.getCode() == KeyCode.RIGHT){
544                 PictureRenderer.inputHandler(INPUT_RIGHT, qualityMap.get(qualityLvl));
545             }
546         }
547     }
```

Meghívja az **inputHandlert**.. ahol összefutnak azaz ami mindig meghívódik:

PictureRenderer/makeImage:

```
256     private static ArrayList<ImageV2> makeImage(@NotNull Constant size, int minX, int minY){
257         ArrayList<ImageV2> newList = new ArrayList<>();
258         switch (size) {
259             case SIZE_1 :
260                 newList.add(new ImageV2(Objects.requireNonNull(renderImage(minX, minY,  imageSize: 1, TYPE_1X1))));
261                 break;
262             case SIZE_4 :
263                 newList.add(new ImageV2(Objects.requireNonNull(renderImage(minX      , minY ,  imageSize: 2, TYPE_2X2))));
264                 newList.add(new ImageV2(Objects.requireNonNull(renderImage( minX: minX + 1 , minY ,  imageSize: 2, TYPE_2X2))));
265                 newList.add(new ImageV2(Objects.requireNonNull(renderImage(minX  , minY: minY + 1,  imageSize: 2, TYPE_2X2))));
266                 newList.add(new ImageV2(Objects.requireNonNull(renderImage( minX: minX + 1 ,  minY: minY + 1 ,  imageSize: 2, TYPE_2X2))));
267                 break;
268             case SIZE_16 :
269                 for (int j = 0; j < 4; j++) {
270                     for (int i = 0; i < 4; i++) {
271                         newList.add(new ImageV2(Objects.requireNonNull(renderImage( minX: minX + i ,  minY: minY + j ,  imageSize: 4, TYPE_4X4))));
272                     }
273                 }
274                 break;
275             case SIZE_64 :
276                 for (int j = 0; j < 8; j++) {
277                     for (int i = 0; i < 8; i++) {
278                         newList.add(new ImageV2(Objects.requireNonNull(renderImage( minX: minX + i,  minY: minY + j,  imageSize: 8, TYPE_8X8))));
279                     }
280                 }
281                 break;
282             case SIZE_256 :
283                 for (int j = 0; j < 16; j++) {
284                     for (int i = 0; i < 16; i++) {
285                         newList.add(new ImageV2(Objects.requireNonNull(renderImage( minX: minX + i,  minY: minY + j,  imageSize: 16, TYPE_16X16))));
286                     }
287                 }
288                 break;
289
290         }
291         return newList;
292     }
```

Kérdés hogy canvast akkor készítesz e ha új négyesre mozog meg vagy akkor ha elkezd rajzolni. Az előbbi egyszerűbb az utóbbi optimalizáltabb.

Ha az előbbit csinálod alszeg ide érdemes betenni a canvas meghívását vagy készítését.

innen csak a **case SIZE_1** érdekel téged mert az a raw image.

A canvas meghívható vagy készíthető az a lapján hogy a **minX, minY**

coordinátákkal létezik e már canvas. Ez azt okozza hogy minden különböző négyesre különböző canvast keres, ahhoz hogy olyan canvast is meg tudj nyitni ami csak félig szerepelne a képen valami okosat kell kitalálnod de szerintem ez most nem prioritás, ha az azonos négyesnél megnyitja az elég egyelőre.

- Nagyítás:

A **makeGridVieweren** belül található a basic képlet ide egyelőre elég talán annyi hogy a canvas is mozogjon vele együtt és nagyítson de rajzolni ne lehessen.

```
456         stackPane.setOnScroll(event -> {
457             if (isZoomReady){
458                 if (event.getDeltaY() < 0){
459                     //Zoom out
460                     if (grid.getScaleX() > 1){
461                         grid.setScaleX(grid.getScaleX() / 2);
462                         grid.setScaleY(grid.getScaleX());
463                     }
464                 }
465                 else {
466                     //Zoom in
467                     if (grid.getScaleX() < 100){
468                         grid.setScaleX(grid.getScaleX() * 2);
469                         grid.setScaleY(grid.getScaleX());
470                     }
471                 }
472             }
473         });
474         stackPane.setOnMousePressed(event -> {
475             if (isZoomReady) {
476                 orgSceneX = event.getSceneX();
477                 orgSceneY = event.getSceneY();
478                 orgTranslateX = grid.getTranslateX();
479                 orgTranslateY = grid.getTranslateY();
480             }
481         });
482         stackPane.setOnMouseDragged(event -> {
483             if (isZoomReady) {
484                 double offsetX = event.getSceneX() - orgSceneX;
485                 double offsetY = event.getSceneY() - orgSceneY;
486                 double newTranslateX = orgTranslateX + offsetX;
487                 double newTranslateY = orgTranslateY + offsetY;
488
489                 grid.setTranslateX(newTranslateX);
490                 grid.setTranslateY(newTranslateY);
491             }
492         });
```

- Kimentés:

Az eddigiek alapján, a **FileStructureben** a **makeImage** adja vissza a kimentedő kép helyét míg a **makeCanvasFile** adja vissza a maszk helyét (és a makeImage által visszaadott fájlt kéri)

```
    @NotNull
    @Contract("_ -> new")
    static File makeFile(@NotNull SnappedImage snappedImage){
        String sequenceID = snappedImage.getId();
        analyseSequenceID(sequenceID);
        File secondaryFile = makeDirectory(snappedImage.getFlag());
        fileLogger(snappedImage);
        folderLogger();
        String end = snappedImage.getFlag().equals("Positive") ? "_pos" : "_neg";
        assert secondaryFile != null;
        return new File(pathFormatter( ...strings: secondaryFile.getPath(),folderOneName,folderTwoName,sequenceID + end + snapExtension));
    }

    @NotNull
    @Contract("_ -> new")
    static File makeCanvasFile(@NotNull File snappedFile){
        return new File( pathname: snappedFile.getParent() + "\\MASK_" + snappedFile.getName());
    }
```

Ha canvast is mentessz ki azt érdemes a raw image folderbe tenni amit az alábbiak alapján talász meg:

```
public class ThumbnailCreationController {

    private static long orgTime = (long) 0;
    private static Time elapsedTime = new Time(0);

    private static Timeline waitAnimation = new Timeline();
    private static Timeline stopperAnimation = new Timeline();

    private static File imageFolder = new File( pathname: projectFolder + "\\" + settings.get( sectionName: "ROBOT_FUNCTIONS", optionName: "RAW_IMAGE_FOLDER_FULL_NAME") + "\\");
    static int imageListSize = 0;
```

|||

VVV

```
private static File imageFolder = new File( pathname: projectFolder + "\\" + settings.get( sectionName: "ROBOT_FUNCTIONS", optionName: "RAW_IMAGE_FOLDER_FULL_NAME") + "\\");
```

Ez a folder már inicializálva lesz amikor a képek meg vannak jelenítve tehát ebbe pakolhatsz nyugodtan, vagy csinálhatsz külön foldert is a lényeg hogy érdemes a **minX, minY** coordináták alapján névvel kimenteni hogy utána meg tudd hívni.

Egyéb megjegyzés:

- A kimentő program neve **PanoramicViewer,** ha ezt letöltöd és amikor futtatod a programot és rámész hogy new project majd browse, kiválasztod a panoramic viewer MView.exe-jét akkor már megvan a program csatolás
- Képet úgy kapsz hogy átküldöm a harangis e-mailt a netes képek belépéséről, ha már felraktad a progit ott megnyitsz egy képet és a programmal fogja megnyitni aztán bal oldalt **File -> Save slide as…** és kiválasztod hova és aztán a szakdogában ha már rámész a **new Project**-re akkor felhozza hogy kiválaszd a fájlt, azt kiválasztod (.mrxs) majd adsz neki egy nevet és mented és olyan kor nem nyomsz semmit, elindul a panoramic viewer, elkezdi

nemsokára kimenteni majd ha az kész elkezdi convertálni és ha az is megvan akkor kész vagy és tesztelheted

- Érdemes minél hamarabb ezt megcsinálni mert hosszú idő, közben tesztelheted hogy hogyan csináld meg az egyéb részeket.
- **Egyéb egyéb**: a **MainPageController/isZoomReady** egy olyan boolean ami akkor ad igazat vissza amikor a raw image van megjelenítve tehát ezt használhatod arra hogy nézd nehogy a több képes megjelenítés legyen éppen.