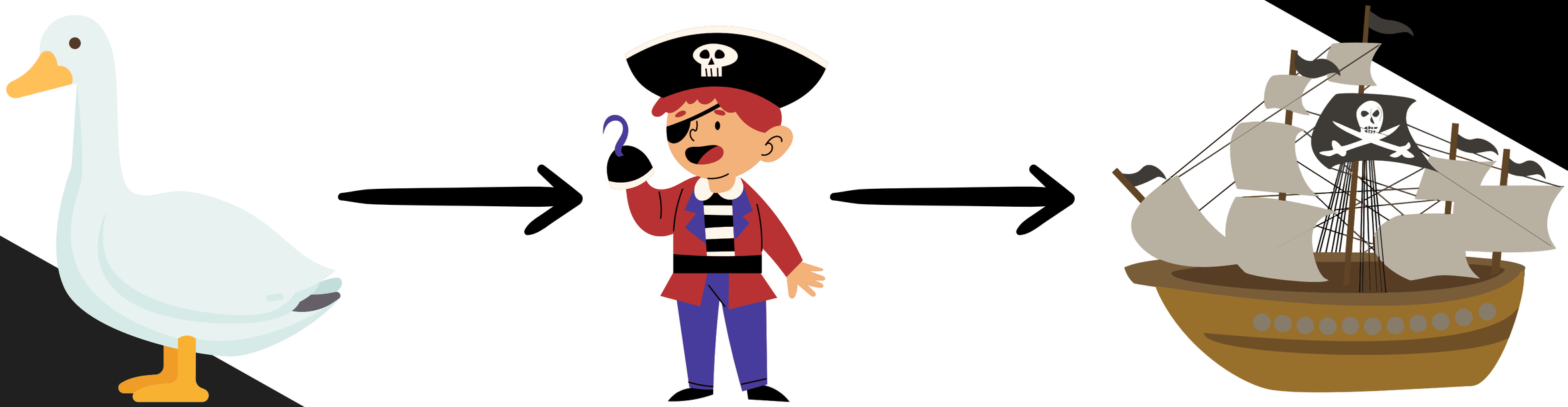
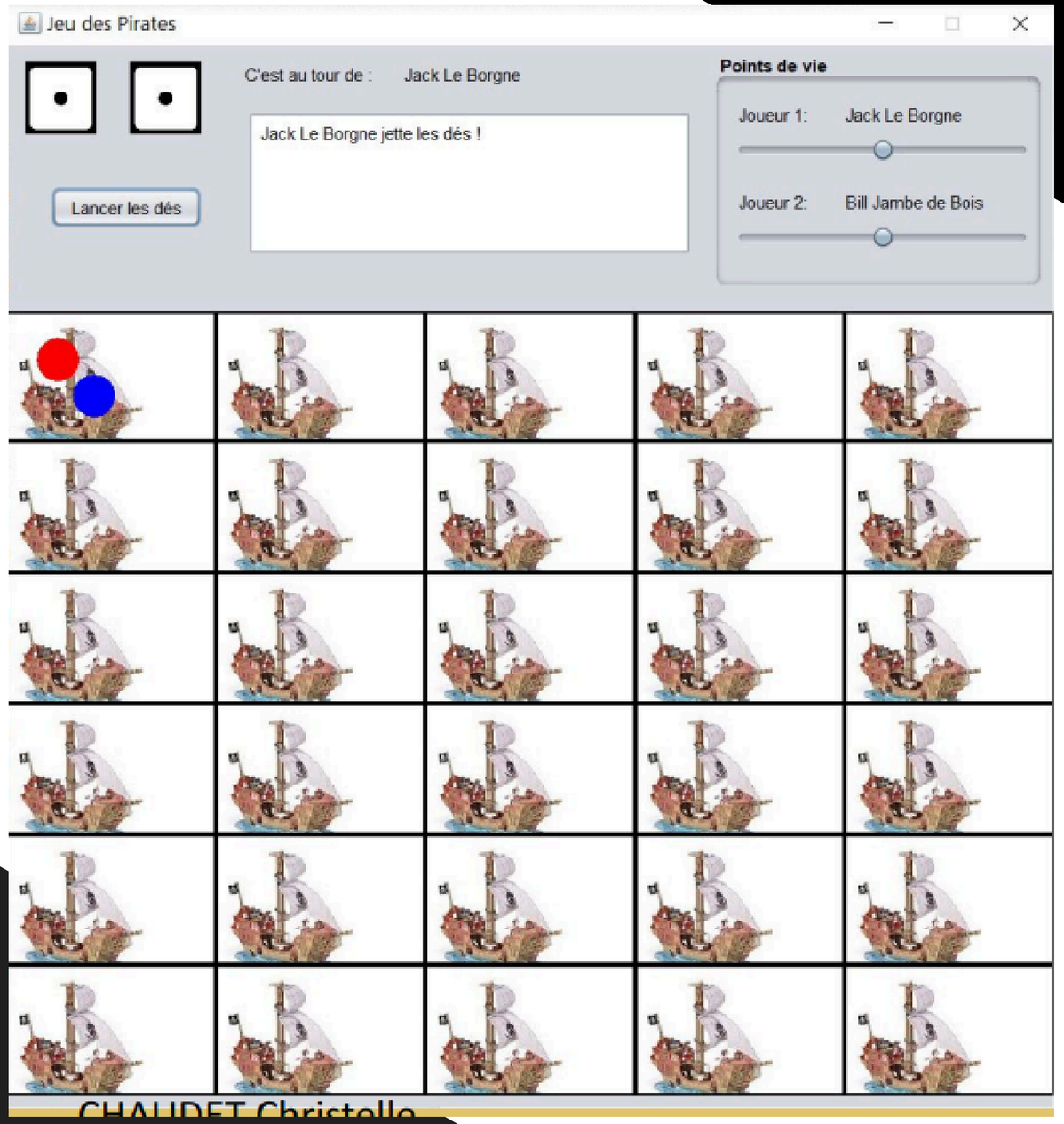
PROJET : LES PIRATES

BEN JAAFAR CHEDLI /RIBEIRO DUARTE
BAGDATLI OKTAY / MUNOS ENZO
GUICHARD LUCAS / ESSENGUE MATIS

Vendredi 24 mai 2024

Université Paul Sabatier - 2024



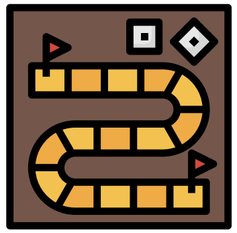

Jeu des Pirates
C'est au tour de : Jack Le Borgne
Jack Le Borgne jette les dés !
Lancer les dés
Points de vie
Joueur 1: Jack Le Borgne
Joueur 2: Bill Jambe de Bois
CHAUDET Christelle

Dés Animées

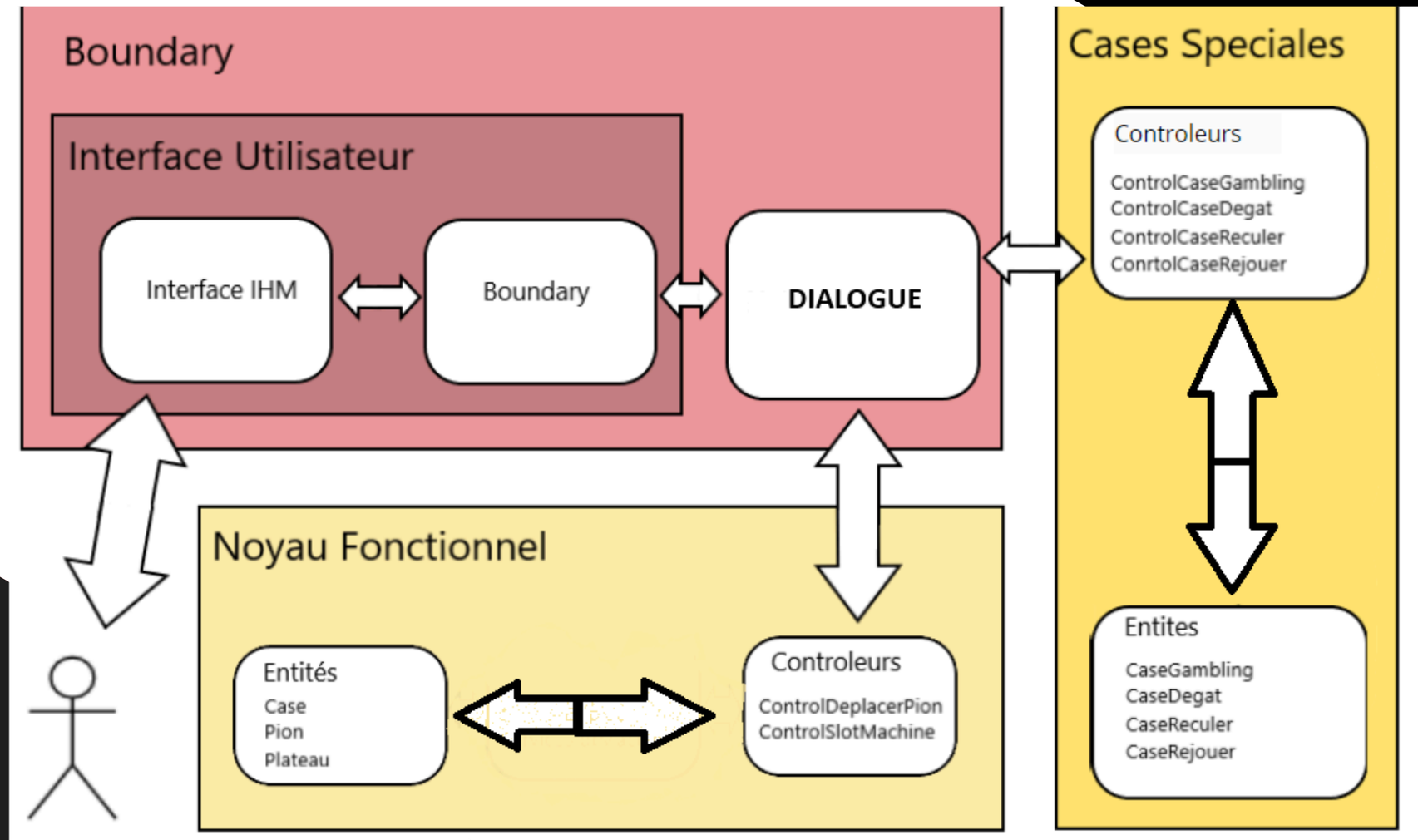
Autre visuel de point de vie

Numéro des cases

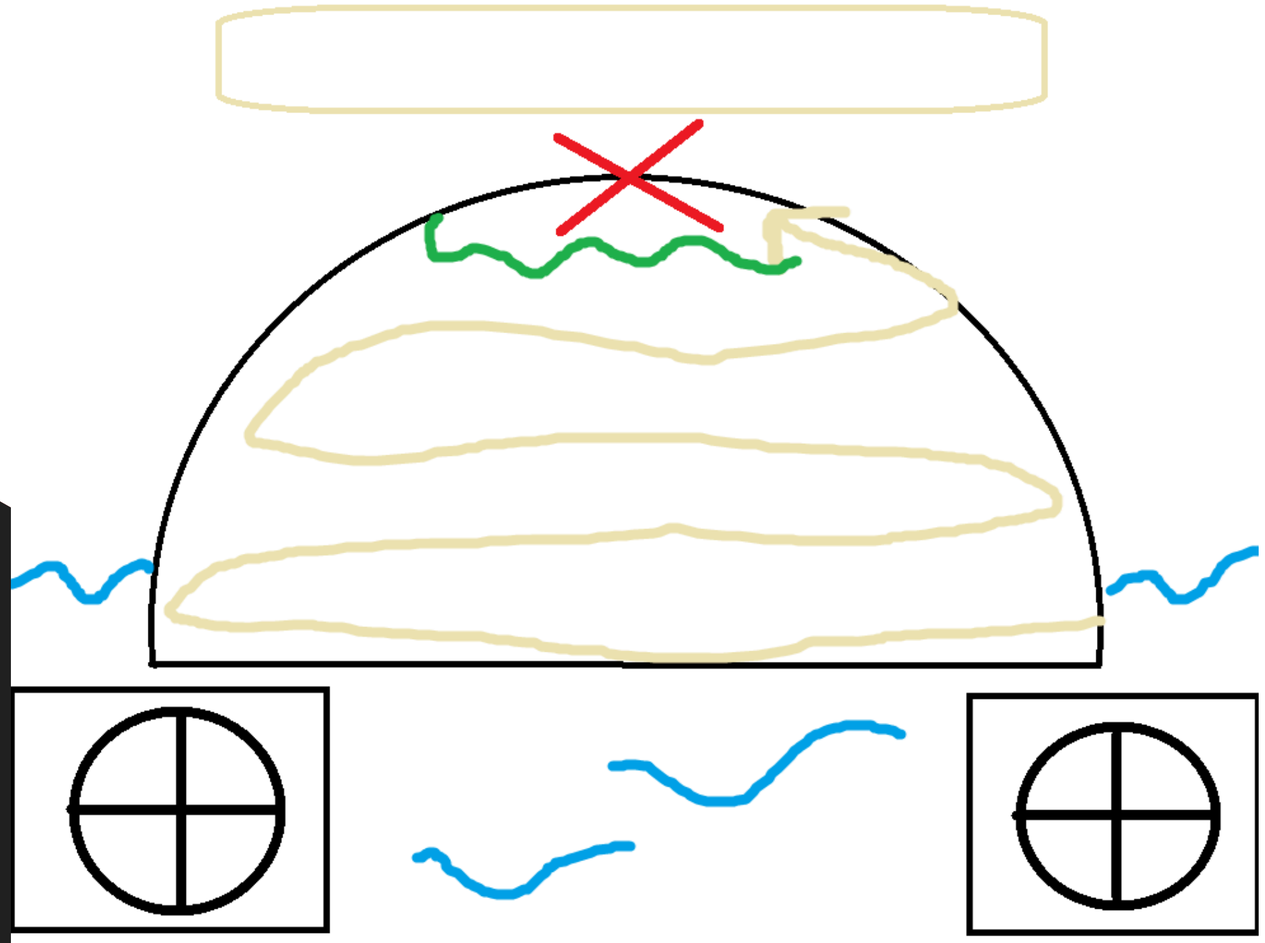
Distinction case départ et arrivée



Des Bateaux ???


Boundary
Interface Utilisateur
Interface IHM
Boundary
DIALOGUE
Cases Speciales
Controleurs
ControlCaseGambling
ControlCaseDegat
ControlCaseReculer
ConrtolCaseRejouer
Noyau Fonctionnel
Entités
Case
Pion
Plateau
Controleurs
ControlDeplacerPion
ControlSlotMachine
Entites
CaseGambling
CaseDegat
CaseReculer
CaseRejouer

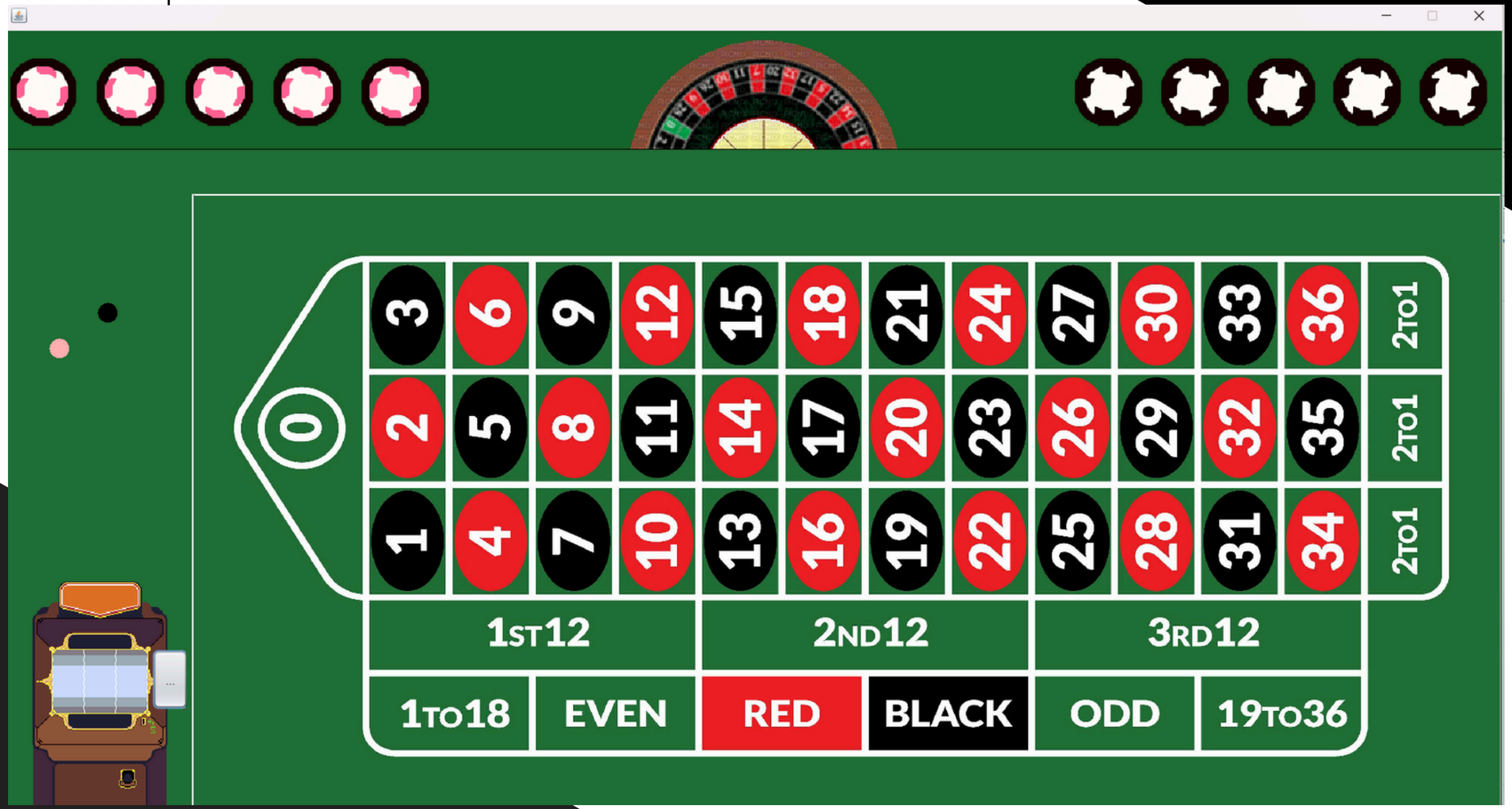
Mockup ecran premières idées



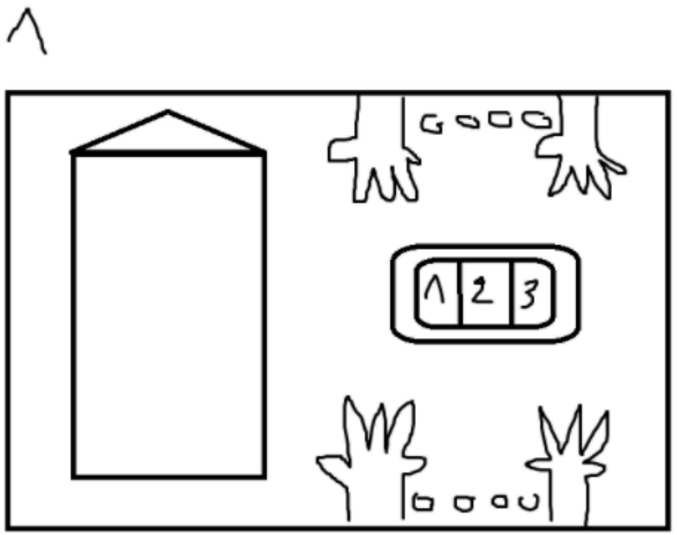
Visuelle Animal Crossing

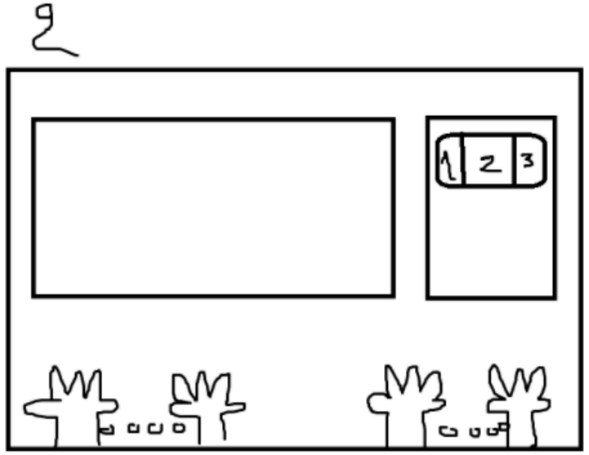
Visuelle Croque-Carotte

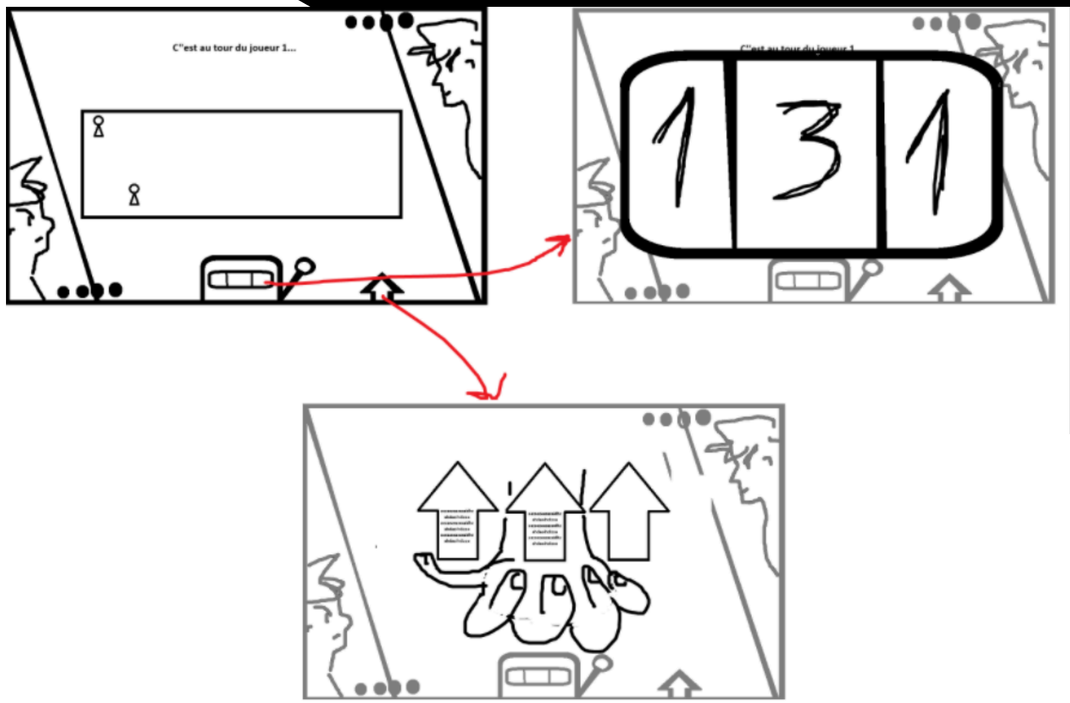
Première implémentation

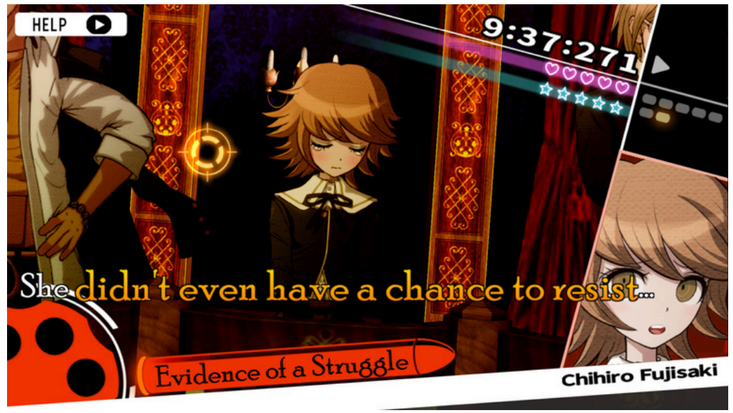


Première axe d’idée

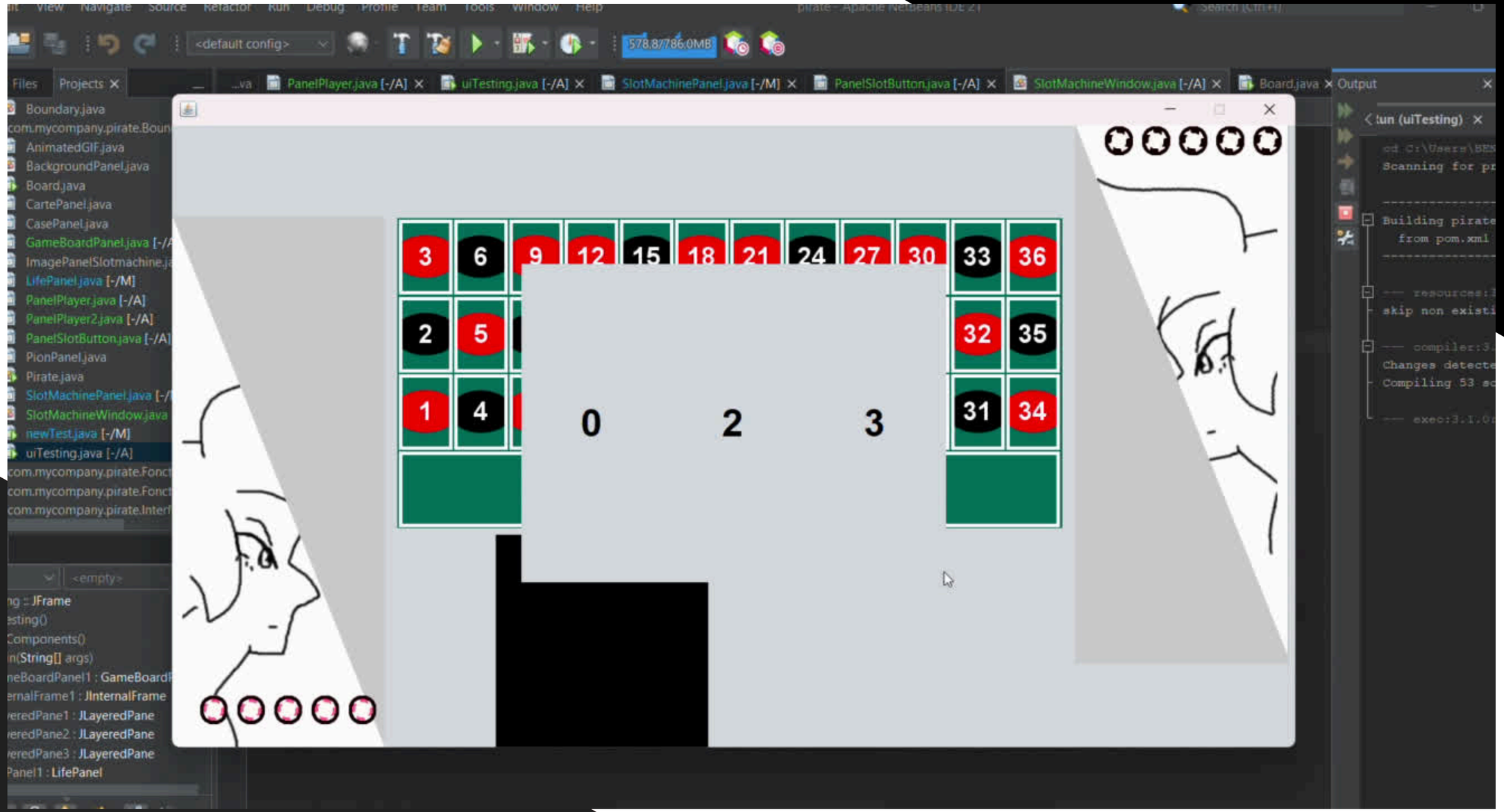




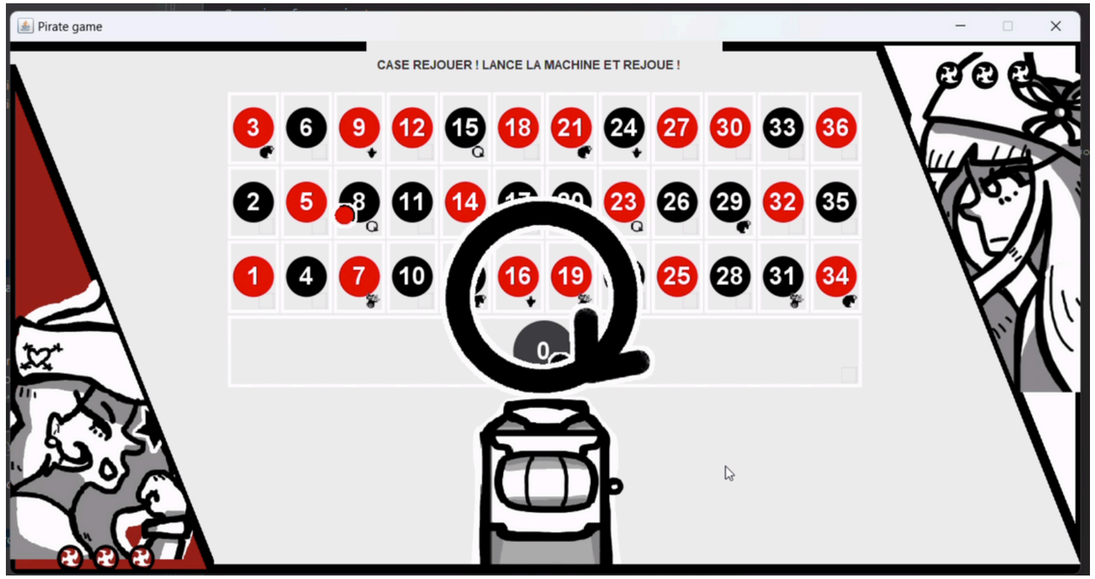


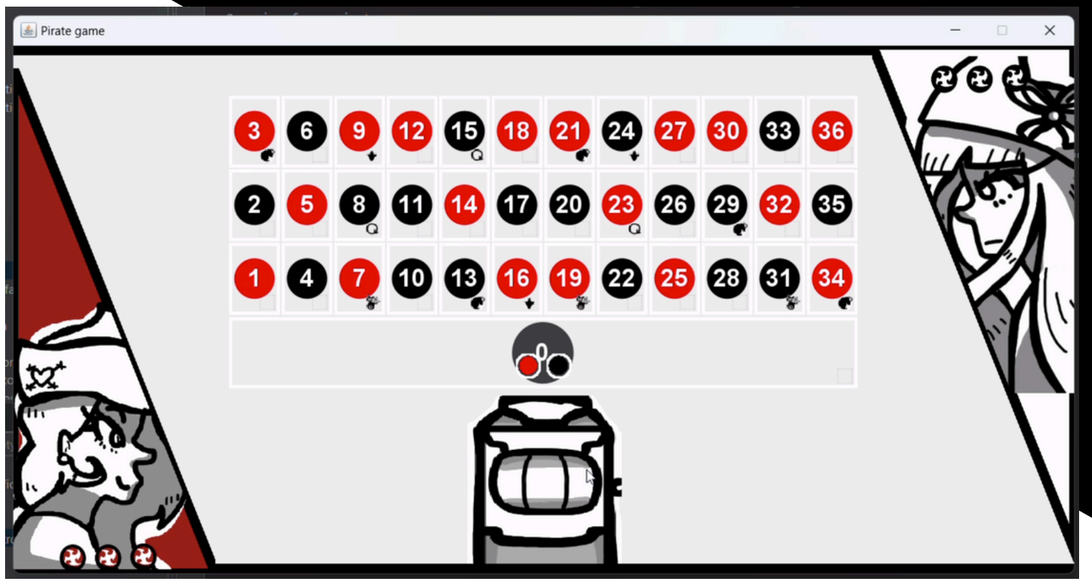


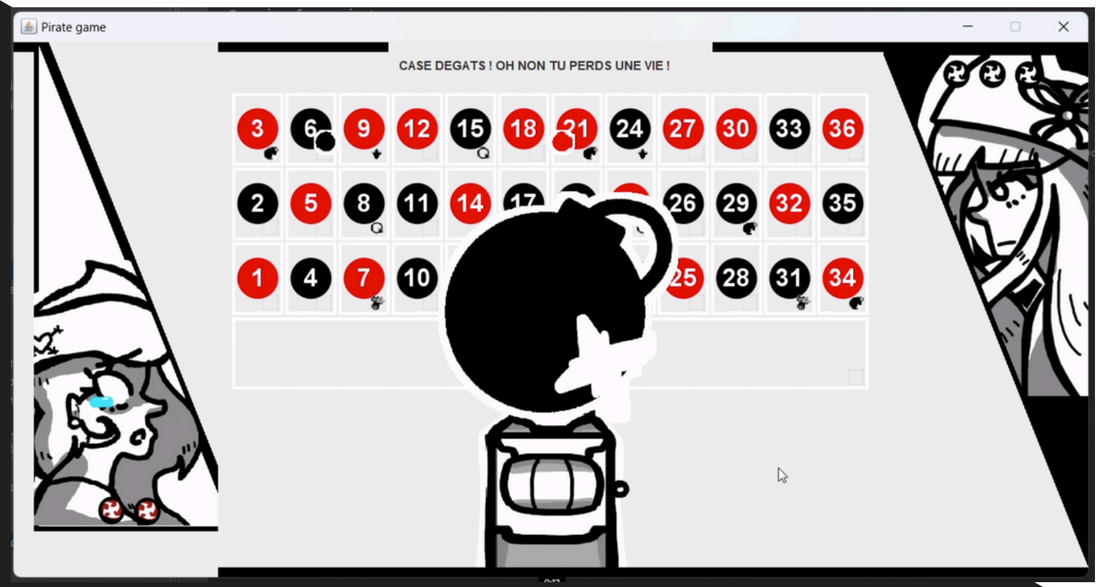
Second axe d’idée

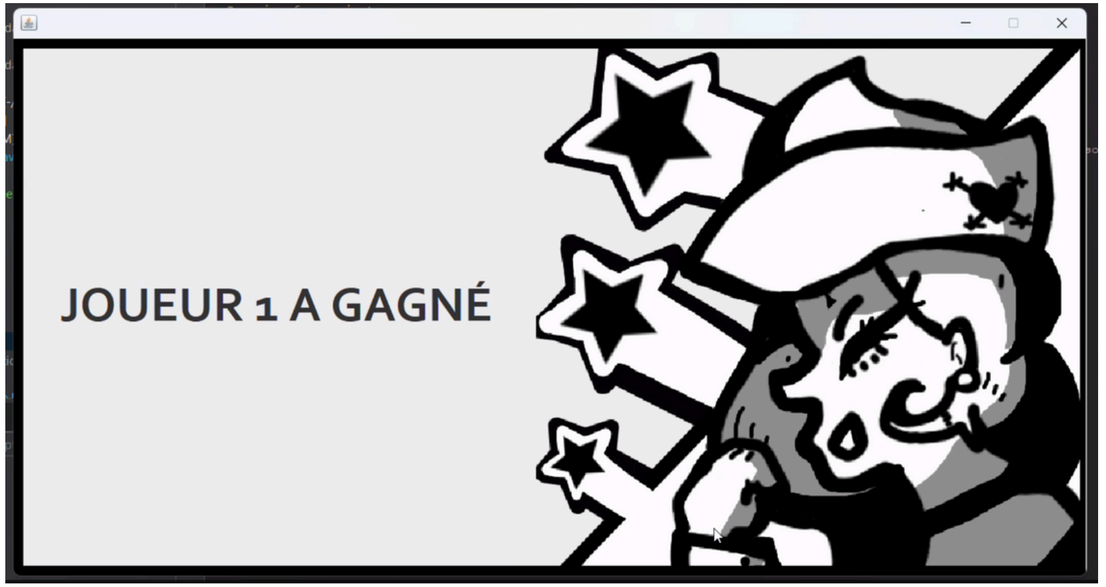

3
6
9
12
15
18
21
24
27
30
33
36
2
5
32
35
1
4
31
34
0
2
3

clideo.com


Pirate game
CASE REJOUER ! LANCE LA MACHINE ET REJOUE !

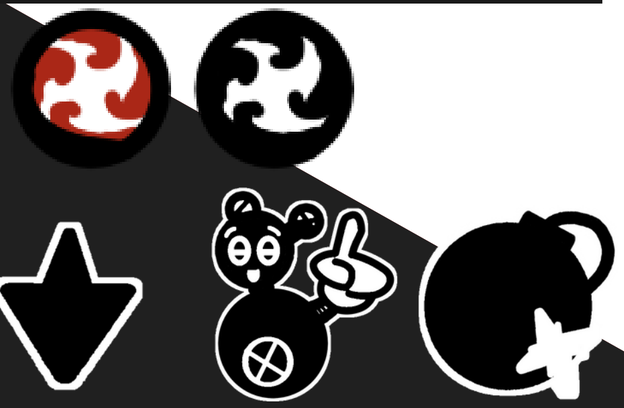

Pirate game


Pirate game
CASE DEGATS ! OH NON TU PERDS UNE VIE !

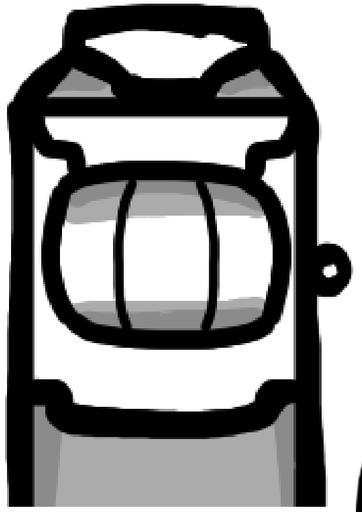

JOUEUR 1 A GAGNÉ

Beaucoup d’effet sonore pour un maximim de retour

Nom	Modifié le	Type	Taille
FX_change_turn.wav	23/05/2024 10:55	Fichier WAV	15 Ko
FX_click.wav	23/05/2024 10:55	Fichier WAV	32 Ko
FX_damage.wav	23/05/2024 10:55	Fichier WAV	179 Ko
FX_gambingmachine_victory.wav	23/05/2024 10:55	Fichier WAV	116 Ko
FX_machine_result.wav	23/05/2024 10:55	Fichier WAV	321 Ko
FX_machine_roulette.wav	23/05/2024 10:55	Fichier WAV	262 Ko
FX_offSlotMachine.wav	23/05/2024 10:55	Fichier WAV	9 Ko
FX_onSlotMachine.wav	23/05/2024 10:55	Fichier WAV	26 Ko
FX_popup_case.wav	20/05/2019 15:48	Fichier WAV	30 Ko
FX_saut.wav	24/05/2024 00:20	Fichier WAV	63 Ko
FX_win.wav	23/05/2024 21:50	Fichier WAV	402 Ko
VOICELINE_player1_damage.wav	24/05/2024 11:12	Fichier WAV	60 Ko
VOICELINE_player1_victory.wav	24/05/2024 11:12	Fichier WAV	52 Ko
VOICELINE_player1_win.wav	23/05/2024 21:59	Fichier WAV	156 Ko
VOICELINE_player2_damage.wav	24/05/2024 11:12	Fichier WAV	47 Ko
VOICELINE_player2_victory.wav	24/05/2024 11:12	Fichier WAV	109 Ko
VOICELINE_player2_win.wav	23/05/2024 21:49	Fichier WAV	238 Ko

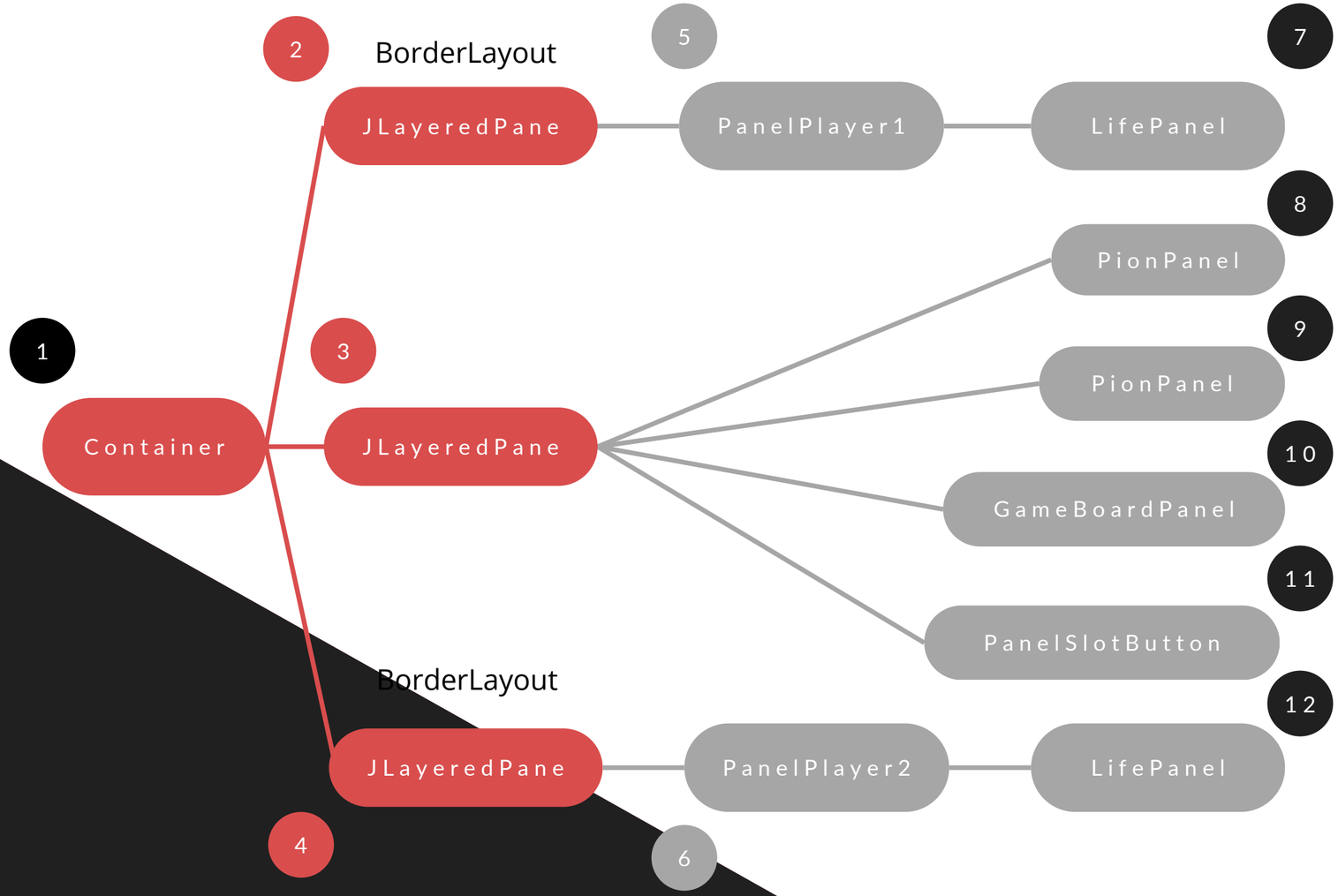


Spirite des icones

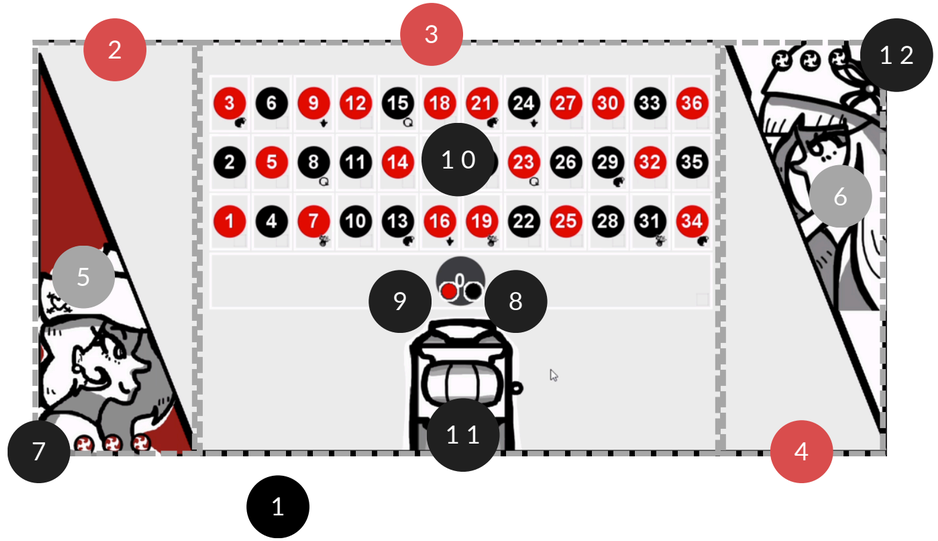


Spirite machine a sous

Sprites des joueurs


1
2
BorderLayout
3
4
5
6
7
8
9
10
11
12
Container
JLayeredPane
JLayeredPane
JLayeredPane
PanelPlayer1
PanelPlayer2
LifePanel
LifePanel
PionPanel
PionPanel
GameBoardPanel
PanelSlotButton
BorderLayout


Form UI - Navigator
Form UI
Other Components
[JFrame]
LayeredPaneMain [JLayeredPane]
PanelPion1 [InvalidComponent]
PanelPion2 [InvalidComponent]
PanelGameboard [InvalidComponent]
PanelButtonSlotMachine [InvalidComponent]
LayeredPanePlayer1 [JLayeredPane]
BorderLayout
PanelDisplayerPlayer1 [InvalidComponent]
LayeredPanePlayer2 [JLayeredPane]
BorderLayout
PanelDisplayPlayer2 [InvalidComponent]


1
2
3
4
5
6
7
8
9
10
11
12

- **PanelPlayerDisplayer**

```java
public abstract class PanelPlayerDisplay extends javax.swing.JPanel {
    protected BufferedImage playerImage;
    protected BufferedImage idleImage;
    protected BufferedImage victoryImage;
    protected BufferedImage damageImage;
    protected int triangleBase = 200;
    protected int triangleHeight = 500;
    protected Color colorBackground = Color.white;


    // Etat de tour
    protected Color turnColor = Color.white;

    public enum PlayerState {
        IDLE,
        VICTORY,
        DAMAGE
    }

    protected PlayerState currentState = PlayerState.IDLE;

    protected void loadImage(String imagePath) {...7 lines }

    protected BufferedImage loadImageFromFile(String path) {...8 lines }

    public void setState(PlayerState state) {...4 lines }

    protected void updateImage() {...8 lines }

    protected abstract void startAnimation();

    public void setTurn(boolean bool) {...8 lines }

    public abstract void playVoiceline(PlayerState state);
}
```

```java
/**
 *
 * @author BEN JAAFAR
 */
public class PanelPlayer1 extends PanelPlayerDisplay {
    private double imageY = 0;
    private double imageSpeed = 0.2;
    private boolean movingDown = true;
    private SoundPlayer voicelineDamage = new SoundPlayer(VOICELINE_PLAYER1_DAMA
    private SoundPlayer voicelineVictory = new SoundPlayer(VOICELINE_PLAYER1_VIC

    public PanelPlayer1() {...10 lines }

    private void loadImages() {...5 lines }

    @Override
    public void startAnimation() {...28 lines }

    @Override
    protected void paintComponent(Graphics g) {...32 lines }

    @Override
    public void playVoiceline(PlayerState state) {...6 lines }

    @SuppressWarnings("unchecked")
    Generated Code
```

- **PanelPlayerDisplayer**

```java
@Override
public void startAnimation() {
    Timer timer = new Timer(20, e -> {
        if (playerImage != null && playerImage.getWidth() != 0) {
            int imageWidth = triangleBase - 20;
            int imageHeight = playerImage.getHeight() != 0 ? (int) ((double) playerImage.getHeight() / playerImage.getWidth() * imageWidth) : 0;

            if (imageHeight > triangleHeight - 20) {
                imageHeight = triangleHeight - 20;
                imageWidth = (int) ((double) playerImage.getWidth() / playerImage.getHeight() * imageHeight);
            }

            if (movingDown) {
                imageY += imageSpeed;
                if (imageY >= triangleHeight - imageHeight) {
                    movingDown = false;
                }
            } else {
                imageY -= imageSpeed;
                if (imageY <= 150) {
                    movingDown = true;
                }
            }

            repaint();
        }
    });
    timer.start();
}
```

- **LifePanel**

```java
/**
 *
 * @author BEN JAAFAR & MUNOS
 *
 */
public class LifePanel extends javax.swing.JPanel {
    private boolean isPlayer1 = true;
    private int viesRestantes = MAX_LIFE;
    private BufferedImage lifeImage;
    private final int lifeImageWidth = 30;
    private final int lifeImageHeight = 30;
    private LifeAnimation[] animations;

    public LifePanel() {...7 lines }

    private void startAnimations() {...7 lines }

    private void loadImage() {...7 lines }

    @Override
    protected void paintComponent(Graphics g) {...20 lines }

    public void perdreVie() {...6 lines }

    public void setPlayer2() {...4 lines }

    private class LifeAnimation extends Thread {...37 lines }

    @SuppressWarnings("unchecked")
    Generated Code
```

```java
private class LifeAnimation extends Thread {
    private int yOffset = 0;
    private boolean running = true;

    public LifeAnimation(int delay) {
        setDaemon(true);
        try {
            Thread.sleep(delay);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public void run() {
        double time = 0.0;
        double amplitude = 5.0;
        double frequency = 0.02;
        while (running) {
            try {
                Thread.sleep(16);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            yOffset = (int) (amplitude * Math.sin(frequency * time));
            time += 1.0;
            repaint();
        }
    }

    public int getYOffset() {
        return yOffset;
    }

    public void stopAnimation() {
        running = false;
    }
}
```

- **GameBoardPanel**

```
 * @author BEN JAAFAR & RIBEIRO
 *
 * Chedli -> conception du composants + creation du board (avec les cellules)
 *           + conception de la methode deplacerPion
 * RIBEIRO -> Animationpath
 * MUNOS -> Afficages des icones pour les cases spéciales
 *
 */
public class GameBoardPanel extends javax.swing.JPanel {
    private static final int PIECE_Y_OFFSET = -50; // Ajustement de la position
    private SoundPlayer fxSaut = new SoundPlayer(FX_SAUT_PION);
    private Map<Integer, ImageIcon> iconMap;

    public GameBoardPanel() {...40 lines }

    private Map<Integer, ImageIcon> loadIcons() {...23 lines }

    public void deplacerPion(PionPanel pion, int destinationCellNumber, Runnable

    private List<Integer> createPath(int start, int end) {...13 lines }

    private void animatePath(PionPanel pion, List<Integer> path, int index, Runn

    private void animatePionMovement(PionPanel pion, int destinationX, int desti

    private class CellPanel extends JPanel {...55 lines }
    @SuppressWarnings("unchecked")
    Generated Code


    // Variables declaration - do not modify
    // End of variables declaration
}
```

```
private class CellPanel extends JPanel {
    private int cellNumber;
    private ImageIcon icon;
    private static final List<Integer> RED_NUMBERS = Arrays.asList(1, 3, 5,
    public CellPanel(int cellNumber, ImageIcon icon) {
        setOpaque(false);
        this.icon = icon;

        this.cellNumber = cellNumber;
        this.setPreferredSize(new Dimension(80, 80)); // Taille préférée pou
    }
    @Override
    protected void paintComponent(Graphics g) {...36 lines }
    private boolean isRed(int number) {
        return RED_NUMBERS.contains(number);
    }
    public int getCellNumber() {
        return cellNumber;
    }
}
@SuppressWarnings("unchecked")
Generated Code
```

- **Système de popup**

```java
/**
 *
 * @author BEN JAAFAR
 */
public class CasePopup extends JWindow {
    private ImageIcon imageIcon;
    protected float alpha = 1f;
    private Timer timer;
    int imageSizeScaled = 250;
    private SoundPlayer popupSound = new SoundPlayer(FX_POPUP_CASE);

    public CasePopup(ImageIcon imageIcon, int displayTime) {...23 lines }

    private void fadeOut() {...15 lines }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        Graphics2D g2d = (Graphics2D) g.create();

        g2d.setComposite(AlphaComposite.SrcOver.derive(alpha));

        Image image = imageIcon.getImage();
        int x = (getWidth() - imageSizeScaled) / 2;
        int y = (getHeight() - imageSizeScaled) / 2;
        g2d.drawImage(image,  x, y, imageSizeScaled, imageSizeScaled, this); //

        g2d.dispose();
    }
}
```

```java
 *
 * @author BEN JAAFAR
 *
 * Permet de ne pas avoir a chager les images a chaque fois et de juste directem
 * afficher la popup, les images etant toujours chargées
 */
public class CasePopupManager {
    private ImageIcon iconBomb;
    private ImageIcon iconRejouer;
    private ImageIcon iconReculer;
    private ImageIcon iconGambi;
    private JWindow textPopup;

    public CasePopupManager() {...6 lines }

    public void popupCaseBomb() {
        new CasePopup(iconBomb, 500);
    }

    public void popupCaseRejouer() {
        new CasePopup(iconRejouer, 1000);
    }

    public void popupCaseReculer() {
        new CasePopup(iconReculer, 1000);
    }

    public void popupCaseGambi() {
        new CasePopup(iconGambi, 1000);
    }

    public void showPopup(JFrame parent, String message) {...27 lines }

    public void closePopup() {...6 lines }
}
```

- **Système de popup**

```
public void showPopup(JFrame parent, String message) {
    textPopup = new JWindow();

    JLabel messageLabel = new JLabel(message);
    messageLabel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

    Dimension labelSize = messageLabel.getPreferredSize();

    textPopup.setSize(labelSize.width + 50, labelSize.height + 10);

    int x = parent.getX() + (parent.getWidth() - textPopup.getWidth()) / 2;

    int y = parent.getY() + parent.getInsets().top;

    textPopup.setLocation(x, y);
    textPopup.add(messageLabel);


    parent.addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosed(WindowEvent e) {
            textPopup.dispose();
        }
    });

    textPopup.setVisible(true);
}
```

- **SlotMachine**

```java
/**
 * @author BEN JAAFAR
 * Panel faisant office de bouton pour lancer la machine a sous
 * Il comprends des animations et fx sonore
 */
public class PanelSlotButton extends javax.swing.JPanel {
    private Image slotMachineImage;
    private SlotMachineWindow window;
    private final SoundPlayer fxOnSlotmachine;
    private final SoundPlayer fxOffSlotmachine;
    private final SoundPlayer fxClick;
    private boolean isMouseOver = false;
    private final MouseAdapter mouseAdapter;

    public PanelSlotButton() {...26 lines }

    private void loadSlotMachineImage() {...3 lines }

    private void handleMouseClick() {...4 lines }

    private void handleMouseEnter() {...6 lines }

    private void handleMouseExit() {...6 lines }

    private void openSlotMachineWindow() {...5 lines }

    public void startAnimation(int[] values, Runnable onAnimationEnd) {...7 lin

    @Override
    protected void paintComponent(Graphics g) {...15 lines }

    public void activateListeners() {...3 lines }

    public void deactivateListeners() {...3 lines }

    @SuppressWarnings("unchecked")
    Generated Code
```

```java
/**
 * @author BEN JAAFAR
 */
class SlotMachineWindow extends JWindow{
    private SoundPlayer fxMachineRoulette = new SoundPlayer(FX_MACHINE_ROULETTE);
    private Runnable endAnimation;

    public SlotMachineWindow() {
        fxMachineRoulette.play();
        fxMachineRoulette.loop();
        setSize(600, 300);

        //On ferme la fenetre apres 3sec
        Timer timer = new Timer(3000, (ActionEvent e) -> {
            dispose();
            fxMachineRoulette.stop();
            fxMachineRoulette.close();
            if(endAnimation != null){
                endAnimation.run();
            }
        });
        timer.setRepeats(false);
        timer.start();
    }


    public void startAnimation(int[] values, Runnable onAnimationEnd) {
        SlotMachinePanel slotMachinePanel = new SlotMachinePanel(values);
        add(slotMachinePanel, BorderLayout.CENTER);
        slotMachinePanel.startAnimation(values);
        this.endAnimation = onAnimationEnd;
    }

}
```

- **SlotMachine**

```
public class SlotMachinePanel extends javax.swing.JPanel {
    private SoundPlayer FX_result = new SoundPlayer(FX_MACHINE_ROULETTE_RESULT);
    private final JLabel[] slotLabels = new JLabel[3];
    private final Random random = new Random();
    private int[] finalValues;
    private final int[] currentValues = new int[3];
    private final int maxValue = 4;
    private static final int ANIMATION_DURATION = 2000;
    private long animationStartTime;
    private Timer timer;

    public SlotMachinePanel(int[] values) {...12 lines }

    public void startAnimation(int[] values) {...5 lines }

    private void updateSlots() {...17 lines }

    @Override
    protected void paintComponent(Graphics g) {...14 lines }
    @SuppressWarnings("unchecked")
    Generated Code


    // Variables declaration - do not modify
    // End of variables declaration
}
```

- **PionPanel & WinFramePanel**

```
 *
 * @author BEN JAAFAR
 */
public class PionPanel extends javax.swing.JPanel {
    private Color pionColor = Color.RED;
    private int cellPosition = 0;
    public int player_number = 0;

    public PionPanel() {
        setOpaque(false);
    }

    @Override
    protected void paintComponent(Graphics g) {...16 lines }

    public void setColor(Color color) {
        this.pionColor = color;
    }

    public void setCellPosition(int cellPos){
        this.cellPosition = cellPos;
    }

    public int getCellPosition(){
        return cellPosition;
    }

    @SuppressWarnings("unchecked")
    Generated Code


    // Variables declaration - do not modify
    // End of variables declaration

}
```

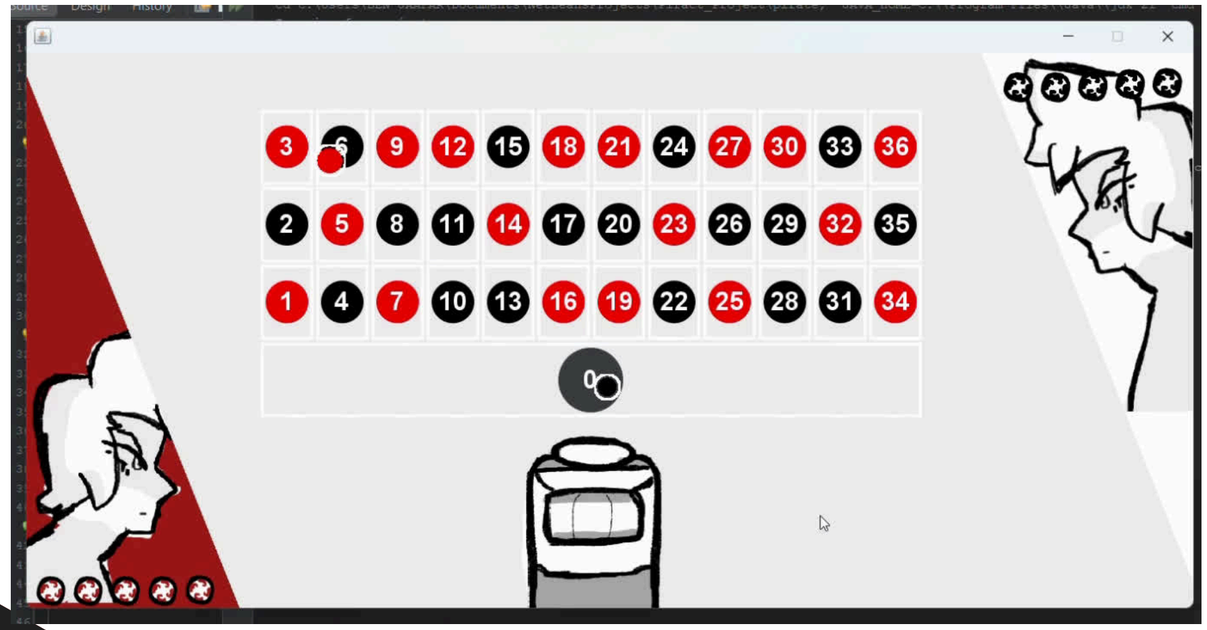
- **PionPanel & WinFramePanel**

```
 * @author BEN JAAFAR
 */
public class FrameWinPlayer extends javax.swing.JFrame {
    private SoundPlayer endingTheme = new SoundPlayer(OST_ENDING);
    private String name;


    public FrameWinPlayer(String name) {
        this.name = name;
        initComponents();
        setLocationRelativeTo(null);
        setResizable(false);
        run();
    }

    private void run() {
        setVisible(true);
        imagePanel1.loadImage(name);
        TestWIn.setText(name.toUpperCase() + " A GAGNÉ");
        endingTheme.setVolume(-10);
        endingTheme.play();
        endingTheme.loop();
    }

    /**
```

```
 * @author BEN JAAFAR
 */
public class PanelImagePlayerWinScreen extends javax.swing.JPanel {
    private Image image = null;
    private String imagePath = null ;

    private int imageX;
    private int imageY;
    private int targetX;
    private int speed = 40;
    private Timer timer;

    private SoundPlayer voiceline;

    public PanelImagePlayerWinScreen() {
        setOpaque(false);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(image, imageX, imageY, this);
    }

    /*load image and voiceline*/
    public void loadImage(String name){...28 lines }

    private void animateImage() {...11 lines }

    @SuppressWarnings("unchecked")
    Generated Code
```

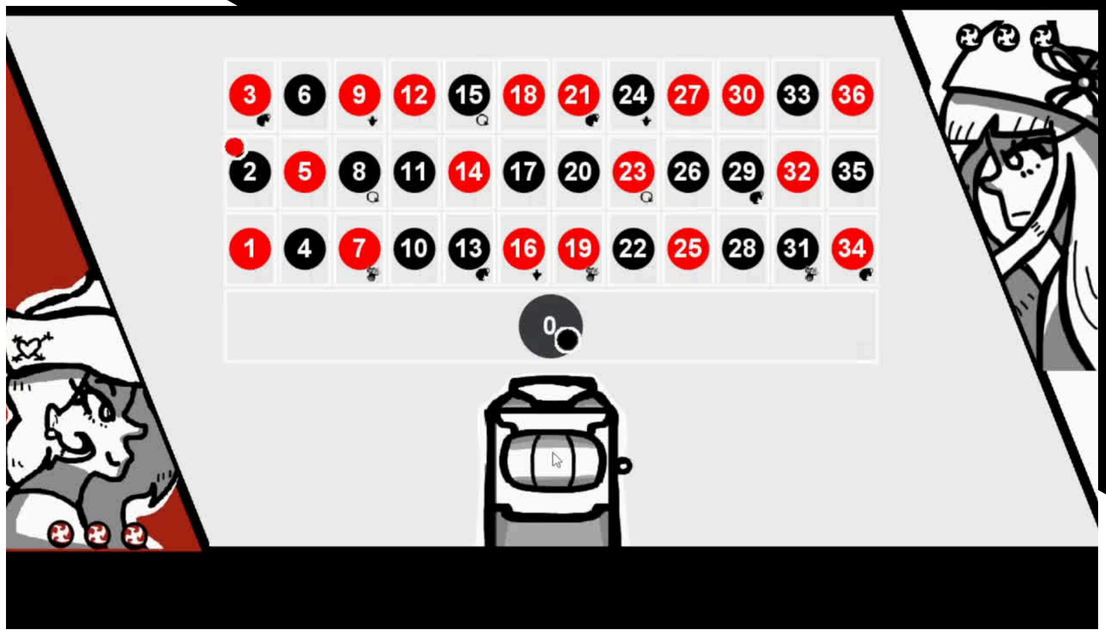
- **Point sur *UI.java***

```java
public void movePiece(int destinationCellNumber, String name) {
    PionPanel currentPlayer = pionPanels.get(name);
    if (currentPlayer != null) {
        int currentIndex = currentPlayer.getCellPosition();
        CountDownLatch latchAnimationEnd = new CountDownLatch(1);
        PanelGameboard.deplacerPion(currentPlayer, currentIndex + destinationCellNumber, () -> latchAnimationEnd.countDown());
        try {
            latchAnimationEnd.await();
            currentPlayer.setCellPosition(currentIndex + destinationCellNumber);
            PanelButtonSlotMachine.activateListeners();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

// Manage the slot machine, its animation, etc.
@Override
public void spinMachineUI(int[] values) {
    CountDownLatch latchClick = new CountDownLatch(1);
    CountDownLatch latchAnimationEnd = new CountDownLatch(1);

    SwingUtilities.invokeLater(() -> {
        try {
            // Code to start the slot machine animation
            setPanelClickListener(() -> latchClick.countDown());
        } catch (Exception e) {
            e.printStackTrace();
        }
    });

    try {
        latchClick.await();
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        return;
```



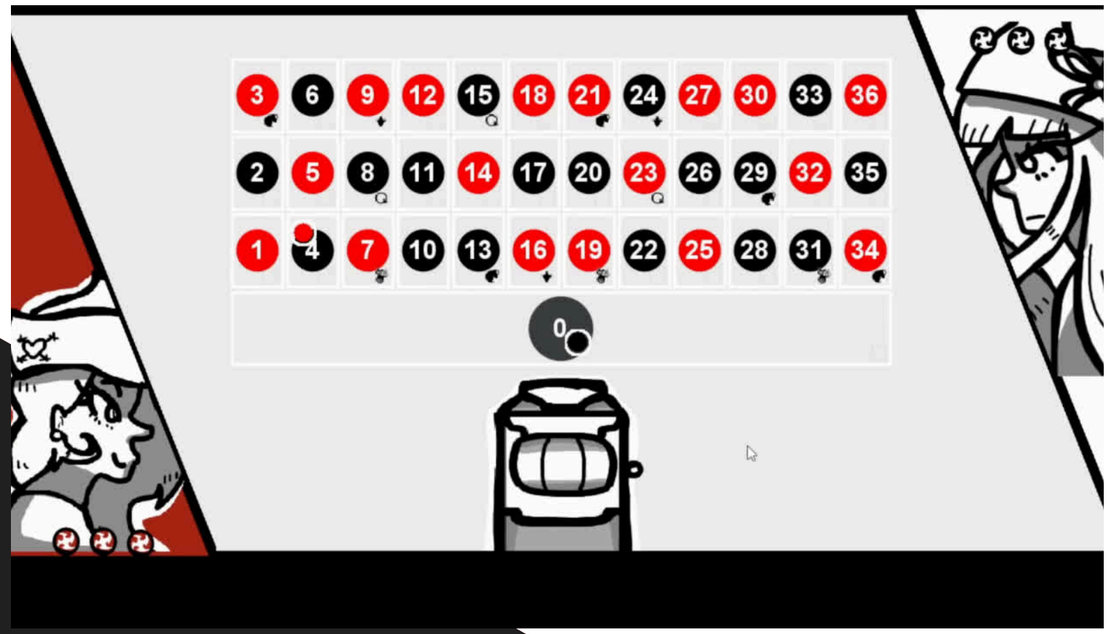

```java
private void animatePionMovement(PionPanel pion, int destinationX, int destinationY, int dest
    int startX = pion.getX();
    int startY = pion.getY();
    int deltaX = destinationX - startX;
    int deltaY = destinationY - startY;
    int steps = 30;
    int delay = 1;
    Timer timer = new Timer(delay, new ActionListener() {
        int step = 0;
        @Override
        public void actionPerformed(ActionEvent e) {
            step++;
            double progress = (double) step / steps;
            int currentX = startX + (int) (deltaX * progress);
            int currentY = startY + (int) (deltaY * progress);
            pion.setLocation(currentX, currentY);
            if (step >= steps) {
                ((Timer) e.getSource()).stop();
                pion.setLocation(destinationX, destinationY);
                pion.setCellPosition(destinationCellNumber);

                // Ajoute un délai d'une seconde ici
                Timer delayTimer = new Timer(500, (ActionEvent ev) -> {
                    if (onAnimationEnd != null) {
                        onAnimationEnd.run();
                    }
                });
                delayTimer.setRepeats(false); // Exécute onAnimationEnd.run() une seule fois
                delayTimer.start();
            }
        }
    });
    timer.start();
```

```java
private void animatePionMovement(PionPanel pion, int destinationX, int destinationY, int destination
    int startX = pion.getX();
    int startY = pion.getY();
    int deltaX = destinationX - startX;
    int deltaY = destinationY - startY;
    int steps = 15; // - de steps = + de vitesse pour le pion
    int delay = 1; // délais entre les steps (1 par defaut pour pas dépasser d'autres evenements)

    Timer timer = new Timer(delay, new ActionListener() {
        int step = 0;
        @Override
        public void actionPerformed(ActionEvent e) {
            step++;
            double progress = (double) step / steps;
            int currentX = startX + (int) (deltaX * progress);
            int currentY = startY + (int) (deltaY * progress);
            pion.setLocation(currentX, currentY);
            if (step >= steps) {
                ((Timer) e.getSource()).stop();
                pion.setLocation(destinationX, destinationY);
                pion.setCellPosition(destinationCellNumber);

                // Ajouter un délai ici si nécessaire
                Timer delayTimer = new Timer(10, (ActionEvent ev) -> {
                    if (onAnimationEnd != null) {
                        onAnimationEnd.run();
                    }
                });
                delayTimer.setRepeats(false); // Exécute onAnimationEnd.run() une seule fois
                delayTimer.start();
            }
        }
    });
```




```java
private void animatePath(PionPanel pion, List<Integer> path, int index, Runnable onAnimationEnd) {
    if (index >= path.size()) {
        if (onAnimationEnd != null) {
            onAnimationEnd.run();
        }
        return;
    }

    int cellNumber = path.get(index);
    for (Component component : getComponents()) {
        if (!(component instanceof CellPanel))
            continue;

        CellPanel cellPanel = (CellPanel) component;
        if (cellPanel.getCellNumber() == cellNumber) {
            int destinationX = cellPanel.getX() + cellPanel.getWidth() / 2;
            int destinationY = cellPanel.getY() + cellPanel.getHeight() / 2 - PIECE_Y_OFFSET;

            if (pion.player_number == 1)
                destinationX -= 15;
            else if (pion.player_number == 2)
                destinationX += 15;

            animatePionMovement(pion, destinationX, destinationY, cellNumber, () -> animatePath(pion, path, index + 1,
            break;
        }
    }
}
```

```
private void animatePionMovement(PionPanel pion, int destinationX, int destinationY, int destinationCellNumber, R
    int startX = pion.getX();
    int startY = pion.getY();
    int deltaX = destinationX - startX;
    int deltaY = destinationY - startY;
    int steps = 20; // - de steps = + de vitesse pour le pion
    int delay = 1; // délais entre les steps (1 par defaut pour pas dépasser d'autres evenements)
    double arcHeightFactor = 0.2; // ajuster la hauteur des sauts

    Timer timer = new Timer(delay, new ActionListener() {
        int step = 0;
        @Override
        public void actionPerformed(ActionEvent e) {
            step++;
            double t = (double) step / steps;
            double arcX = startX + deltaX * t + Math.sin(Math.PI * t) * (Math.abs(deltaY) * arcHeightFactor);
            double arcY = startY + deltaY * t;
            pion.setLocation((int) arcX, (int) arcY);
```



ControllerX implements IDialog

exécute une fonction notify de Boundary à chaque changement d'état dans le jeu

Peuvent se communiquer grace aux fonction notify...

Boundary implements IDialog,IPirates

chauqe fois qu'il reçoit une fonction notify il execute la fonction UI equivalent

avec les fonctions de IPirates

UI implements IPirates

Fait les changements visuels dans l'application

ControllerX

Peuvent se communiquer grace aux appels de fonctions notify...

Boundary implements IDialogue,IPirates

```java
private final IControlSlotMachine controlSlotMachine;
private Random random = new Random();

public ControlGamblingDuel(ControlSlotMachine controlSlotMachine) {
    this.controlSlotMachine = controlSlotMachine;
}

@Override
public int duelDeDes(Pion pion, IDialogue notificationServices) {
    int min = 2;
    int max = 11;
    int randomValue = min + this.random.nextInt(max - min + 1);
    Optional.ofNullable(notificationServices).ifPresent(service -> service.notifyCaseGambling(randomValue));

    //Lancers de dés
    int[] valeurs = spin();
    int res = Arrays.stream(valeurs).sum();


    //perdant perd une vie
    if (res < randomValue) {
        pion.setVie(pion.getVie() - 1);
        Optional.ofNullable(notificationServices).ifPresent(service -> service.notify(pion.getName() + " a perdu le gambling
        Optional.ofNullable(notificationServices).ifPresent(service -> service.notifyDuelResult(pion.getName(),false));
        return -1;
    } else {
        Optional.ofNullable(notificationServices).ifPresent(service -> service.notify("DUEL REUSSI !"));
            Optional.ofNullable(notificationServices).ifPresent(service -> service.notifyDuelResult(pion.getName(),true));
        return 0;
    }
}
```

Qui se trouvent dans

```java
 *
 * @author BEN JAAFAR
 *
 * Adaptateur qui permet le dialogue entre le jeu et la bounda
 */
public interface IDialogue {
    void notify(String message);

    void notifySpin(int[] values);

    void notifyEtatJeu();

    void notifyCaseDegat(String name, int vie);

    void notifyCaseRejouer(String name);

    void notifyCaseReculer();

    void notifyCaseGambling(int randomValue);

    void notifyDuelResult(String name, boolean win);

    void notifyDeplacerPion(int deplacement, String name);

    void notifyNouveauTour(String name);

    void notifyFinDeJeu();


}
```

Boundary implements IPirates,IDialogue

partagent un ensemble des fonctions

UI implements IPirates

Boundary.java

```java
@Override
public void notifyNouveauTour(String name){
    newTurn();
}
@Override
public void newTurn() {
    this.GUI.newTurn();
}
```

Qui se trouve dans

UI.java

```java
public void newTurn() {
    PanelDisplayerPlayer1.setTurn(testingTurn);
    PanelDisplayPlayer2.setTurn(!testingTurn);
    testingTurn = !testingTurn;
    changeTurn.play();
}
```

```java
public interface IPirates {

    public void movePiece(int deplacement, String name);
    public void startGUI();

    public void spinMachine(int[] values);
    public void newTurn();
    public void endGame();
    public void takeDamage(String name);
    public void gamblingDuelResult(String name, boolean win);
    public void caseBombe();
    public void caseRejouer(String name);
    public void caseReculer();
    public void caseGambling(int value);

}
```

Conclusion

BEN JAAFAR CHEDLI /RIBEIRO DUARTE /BAGDATLI OKTAY / MUNOS ENZO / GUICHARD LUCAS / ESSENGUE MATIS

Merci.

PROJET PIRATES - 2024 - UNIVERSITE PAUL SABATIER