Instructions générales

Suivez attentivement les instructions suivantes. Tout manquement sera sanctionné d'un point négatif.
Créez un nouveau projet en langage Java, nommé selon le schéma "TP7_numéro", où "numéro" est votre numéro d'étudiant (sans le 'u' qui le précède).
Si un morceau de votre code ne compile pas, commentez-le avant de déposer votre rendu sur Eprel : un projet qui ne compile pas recevera 0.

Une fois le projet terminé, exportez votre projet (`File` → `Export` → `Export to Zip File`), et rendez ce fichier sur Eprel. N'archivez pas manuellement le dossier contenant votre code !

L'utilisation d'une IA générative et le partage de code entre étudiants sont passibles d'envoi devant le conseil de discipline.

Présentation du jeu *Cobra vs. Mangouste*

Nous allons implémenter un jeu dans lequel le but sera d'amener un cobra d'un point à un autre, en tentant d'éviter au maximum les mangoustes qui se tapissent en embuscade [1].

1 Mise en place du terrain

Question 1 Ajoutez l'image `mangouste.png` à votre projet Android Studio en tant que ressource, en ouvrant l'onglet "ressource manager" dans la barre gauche, et en suivant la procédure illustrée en Figure 1.

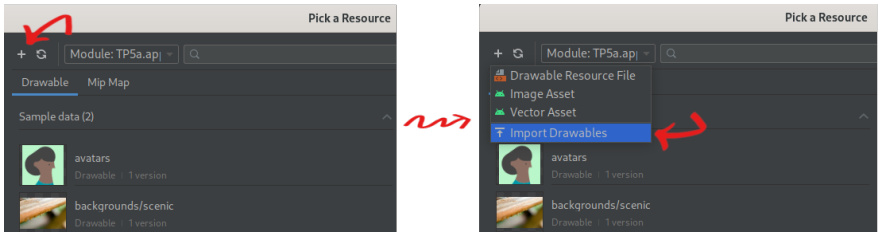


FIGURE 1 – Ajouter une image au projet.

Question 2 Créez une classe `Point` permettant de représenter des points en deux dimensions (avec des coordonnées `float`), avec les attributs, getters et setters nécessaires.

Question 3 Créez une classe `Terrain` héritant de `ConstraintLayout`. Un `Terrain` aura les attributs suivants :

— une `ArrayList` de `Point`, nommée `positions`

1. Les mangoustes sont l'un des rares prédateurs des cobras ; en cas de confrontation, la mangouste sort victorieuse autour de 80% du temps.

— une `ArrayList` de `ImageView`, nommée `mangoustes`
— un `Paint` nommé `paint`
— un entier `dim`, qui servira à stocker les dimensions des images
— un `Random` nommé `random` (on importera `java.util.Random`)

Question 4 Dotez `Terrain` d'un constructueur acceptant un `Context`, en recopiant le code ci-dessous.

```
public Terrain(Context context) {
    super(context);
    setWillNotDraw(false);

    DisplayMetrics dm = getResources().getDisplayMetrics();
    dim = (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,100,dm);

     [...]
}
```

On instanciera également `paint`, `random`, `positions` et `mangoustes` à la place des [...].

Voici quelques explications concernant cette ébauche de constructeur :
— avec `setWillNotDraw(false)`, on demande au `Terrain` de se dessiner (ce n'est pas le comportement par défaut des `ConstraintLayout`).
— on calcule ensuite `dim`, qui sera la taille (en pixels) des images des mangoustes. Ces deux lignes permettent de faire en sorte que les images soient indépendantes de la résolution de l'écran ("DIP" signifie *density-independent pixels*).

Question 5 Pour finir, on va intégrer un `Terrain` au layout de l'`Activity` principale. Pour cela, ajoutez l'instruction suivante à la fin de `onCreate()` :

```
setContentView(new Terrain(this));
```

2 Cobra

On va maintenant se tourner vers le premier élément du jeu : le cobra. Son parcours sera représenté à l'écran par une ligne qui suivra le tracé du doigt à l'écran.

Question 6 Faites en sorte qu'entre le moment où l'utilisateur pose son doigt (pour la première fois) à l'écran, et le moment où il lève son doigt de l'écran, le mouvement du doigt soit représenté par une ligne brisée à l'écran, comme illustré en Figure 2.

On va maintenant vérifier la première des conditions de victoire. Pour gagner, le cobra doit partir du coin en haut à gauche de l'écran, et arriver dans le coin en bas à droite.

Pour être plus précis, on considérera qu'une fois que le doigt de l'utilisateur a quitté (pour la première fois) l'écran, le jeu est gagné si et seulement si
— le doigt a été posé dans le rectangle de taille `10% largeur` par `10% longueur` en haut à gauche du `Terrain`, et
— le doigt a été levé dans le rectangle de taille `10% largeur` par `10% longueur` en bas à droite du `Terrain`.

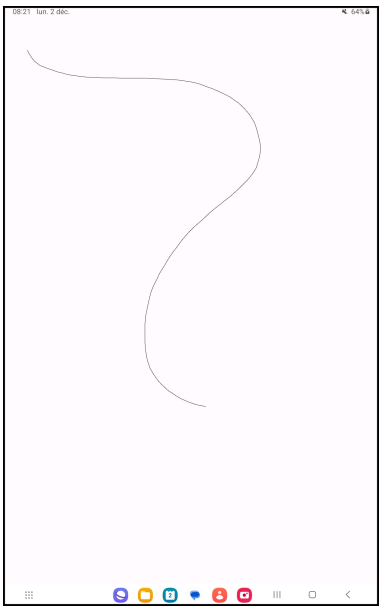


FIGURE 2 – Exemple de trajectoire du cobra : la ligne se dessine en suivant le tracer du doigt.

Des approximations des deux zones en question sont représentées respectivement en vert et en bleu dans la Figure 3a (ces zones de couleur ne doivent pas apparaître sur l'application, elles sont là à titre indicatif). Quelques exemples de victoire et de défaite sont illustrés dans les Figures 3a, 3b et 3c.

Question 7 Implémentez la vérification de la condition de victoire, de sorte qu'après la première fois où le doigt a été levé, un `Toast` "Victoire" ou "Défaite" s'affiche à l'écran, selon que la condition de victoire est vérifiée ou non.
On s'aidera pour ce faire des méthodes suivantes :

- `View::getWidth()`
- `View::getHeight()`

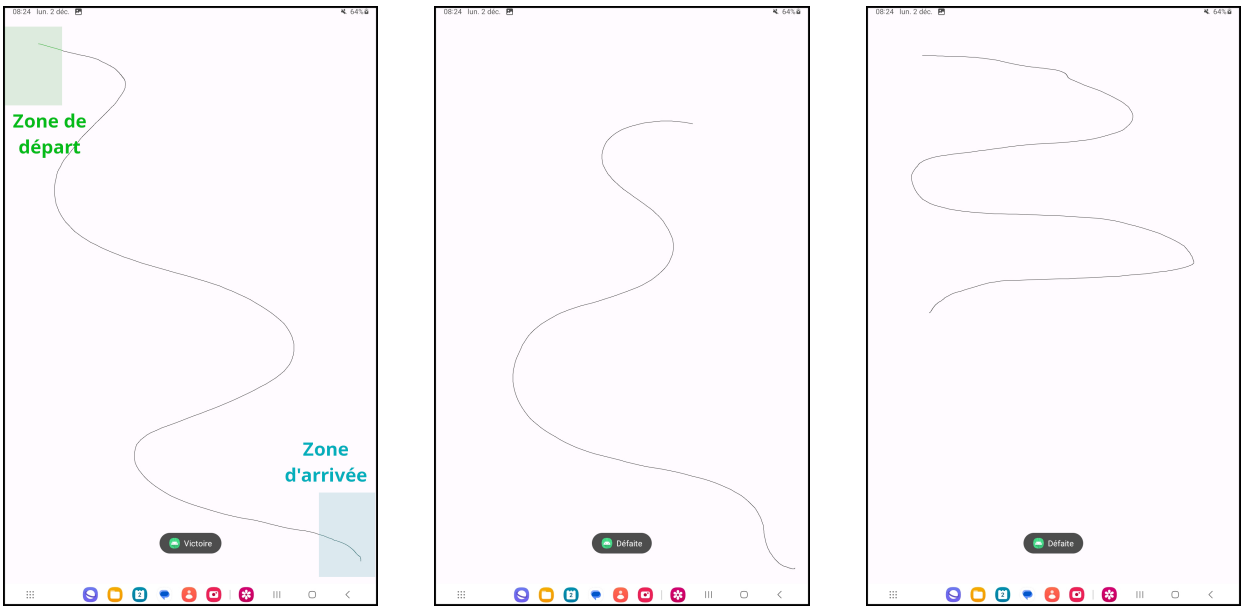
Ces méthodes permettent d'obtenir les dimensions du `Terrain`. On se souviendra qu'il ne faut pas les utiliser dans le constructeur, où le `Terrain` n'a pas encore été placé à l'écran, et donc a des dimensions nulles. On fera donc appel à ces méthodes dans `onTouchEvent()`, où les dimensions du `Terrain` sont connues.

3 Mangoustes

On veut faire apparaitre les mangoustes une fois seulement que le doigt a été posé sur l'écran.

Question 8 En complétant le code de `Terrain`, faites en sorte que lorsque le doigt est posé à l'écran,

1. cinq `ImageView` sont créées et ajoutées à la liste `mangoustes`
2. ces `ImageView` sont ajoutées comme enfants du `Terrain`
3. ces `ImageView` affichent `mangouste.png` à l'aide d'un appel à `ImageView::setBackgroundResource(R.drawable.???)`



(a) Le cobra part bien de sa zone de départ et rejoint sa zone d'arrivée : victoire !

(b) La partie est perdue : le cobra n'est pas parti de sa zone de départ.

(c) La partie est perdue : le cobra s'est arrêté avant de rejoindre sa zone d'arrivée.

FIGURE 3 – Première partie de l'application : gestion du cobra seul.

4. ces `ImagesView` soient des carrés dont les dimensions ont été calculées à la Question 4. Pour cela, on pourra recopier la ligne suivante (pour chaque `ImageView` `view`) :

```
view.setLayoutParams (new Constraints.LayoutParams(dim,dim));
```

5. les positions de positions soit tirées aléatoirement. Pour cela, on fera appel aux méthodes
   — `View::setX()`
   — `View::setY()`
   et à

```
random.nextFloat(); // float entre 0.0 et 1.0
```

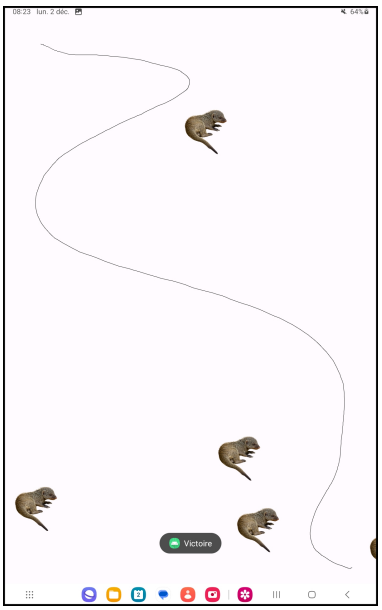
Pour terminer, on va compléter les conditions de victoire pour prendre en compte les mangoustes. Pour remporter la partie, en plus de partir du coin en haut à gauche pour arriver en bas à droite, il faut que la trajectoire évite toutes les mangoustes.

Question 9 Dans `Terrain`, implémentez une méthode

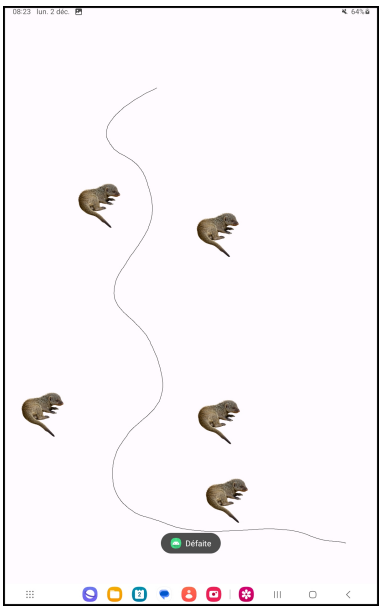
```
boolean isInView(Point p, View v)
```

qui calcule si le `Point` `p` est situé dans la `View` `v`.

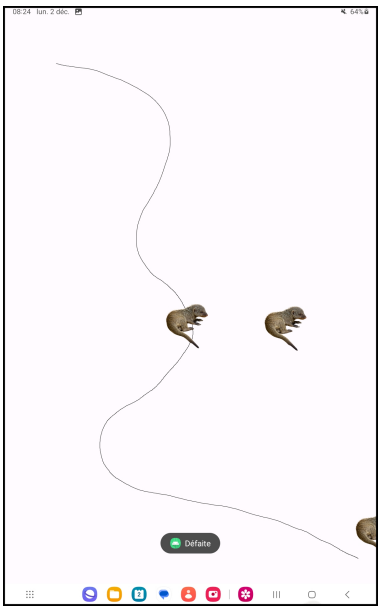
Question 10 Complétez `Terrain::onTouchEvent()` de sorte que, lorsque le doigt est levé, les conditions complètes de victoire soient calculées et le `Toast` adéquat soit affiché. Des exemples sont donnés dans la Figure 4.



(a) La trajectoire du cobra est bonne, et il a évité toutes les mangoustes : victoire !



(b) La partie est perdue : le cobra n'est pas parti de sa zone de départ.



(c) La partie est perdue : le cobra a rencontré une mangouste sur le chemin.

FIGURE 4 – Exemples de condition de victoire avec les mangoustes.