Conception sur OS Nomades

30 Décembre

2015

Application PUZZLE ATTACK

Réalisé par Mohamed Nassim MEZIANI et Toufik SBAIHI

Table des matières

1. Principe de Puzzle Attack

Le concept de l'application Puzzle attaque est très simple vous déplacez des blocs horizontalement autour de l'écran dans un nombre limité de mouvements et de temps pour créer des chaînes de couleurs. Lorsque 3 chaines ou plus de couleurs se connectent, ils disparaissent, et quand tous les blocs ont disparu, vous accéder au niveau supérieur.

Le but est de désactiver l'écran avec un nombre prédéfini de mouvements et dans un laps de temps restreint en mettant en place des réactions d'éliminations en chaîne qui se succèdent.

Puzzle Attaque est un jeu de simple puzzle, mais certaines personnes y deviennent très attachés même « addictif »dans certains cas d'extrémisme.

Le jeu a été décliné en de nombreux thèmes et variantes (publicitaires, thématiques, à licences) et joué sur d'autres supports tels que les téléphones portables.

2. Intérêt du jeu:

Constitue un excellent jeu pour entrainer et tester son intelligence, pratiqué régulièrement, il va permettre d'exercer et de développer la mémoire, le temps de réflexion et la capacité d'analyse et de compréhension des situations complexes et ambigües qui nécessitent des solutions dans des délais restreints.

3. Fonctionnement du jeux

3-1)-Le menu

La classe Menu est MainActivity , qui contient les différentes déclarations et actions des boutons, nos quatre boutons sont:

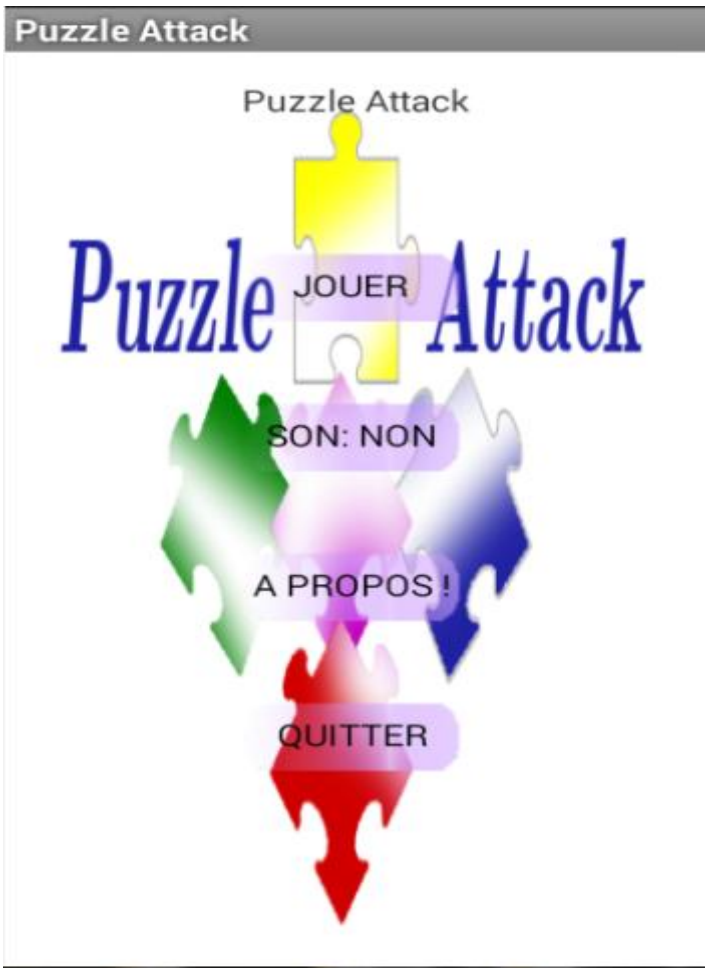


Figure 1: Menu

3-2)-principe du dessin:

Le dessin se fait par la méthode qui est codée comme suit:

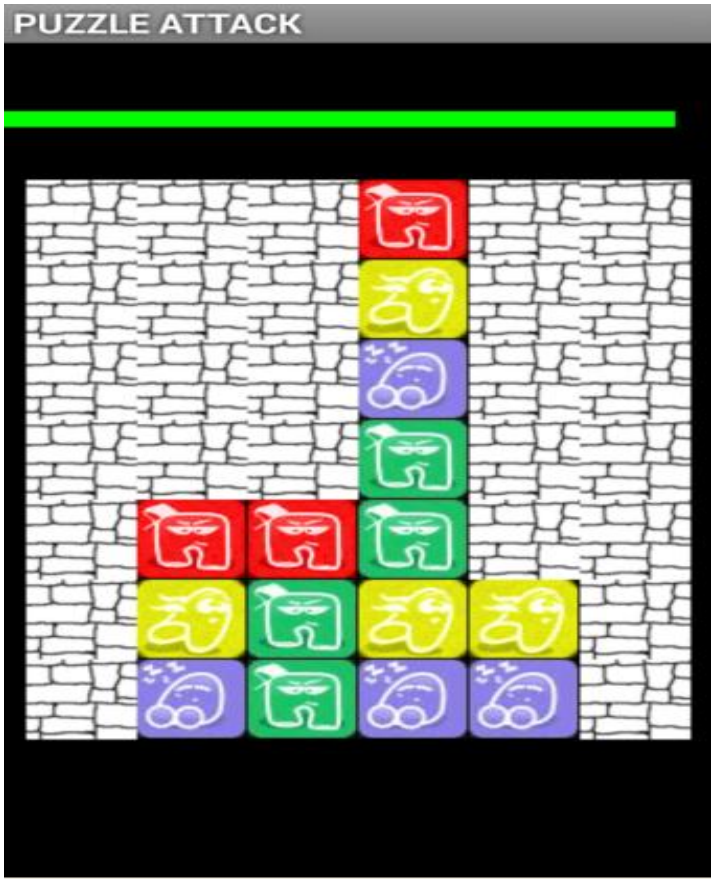
```
private void dessin(Canvas canvas)
     {canvas.drawRGB(0, 0, 0);
     paintMap(canvas);
       paintTimer(canvas);
```

Elle contient deux méthodes:

paintMap: pour le dessin de la matrice, elle colore le fond en noir.

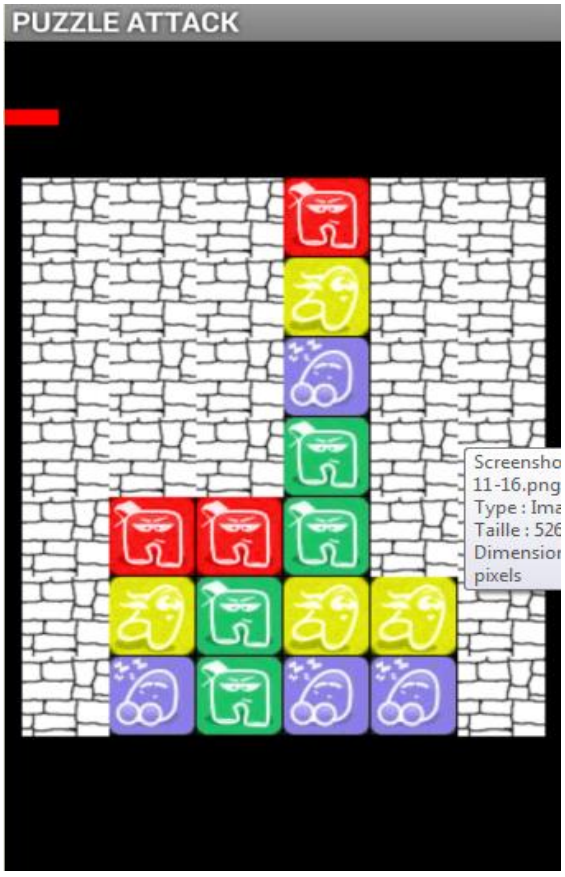
paintTimer: pour dessiner une barre de temps qui est colorée en fonction de la progression du temps.

Au début de la partie la barre est en couleur vert ce qui veut dire que le temps est maximal :

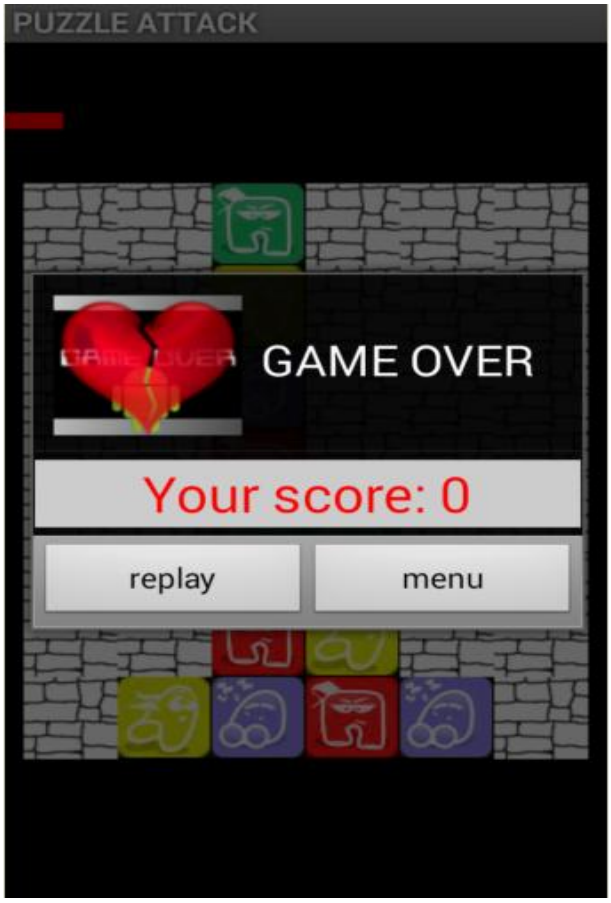


La progression de la barre du temps en jaune :

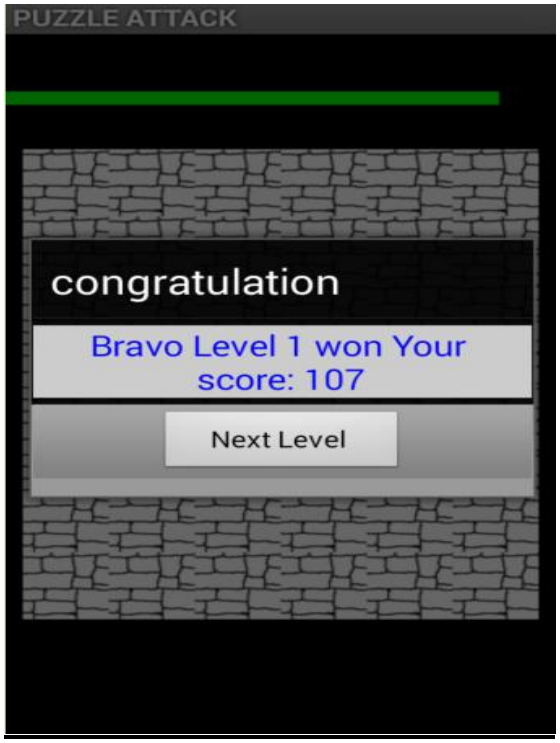
La progression de la barre du temps (rouge) :



Une fois le temps écoulé , un message qui affiche que le jeu est perdu « GAME OVER » et le score sont affichés . Ce message contient deux boutons «replay » et « menu » :
Replay : c'est pour refaire la parti , menu : pour le retour au menu.

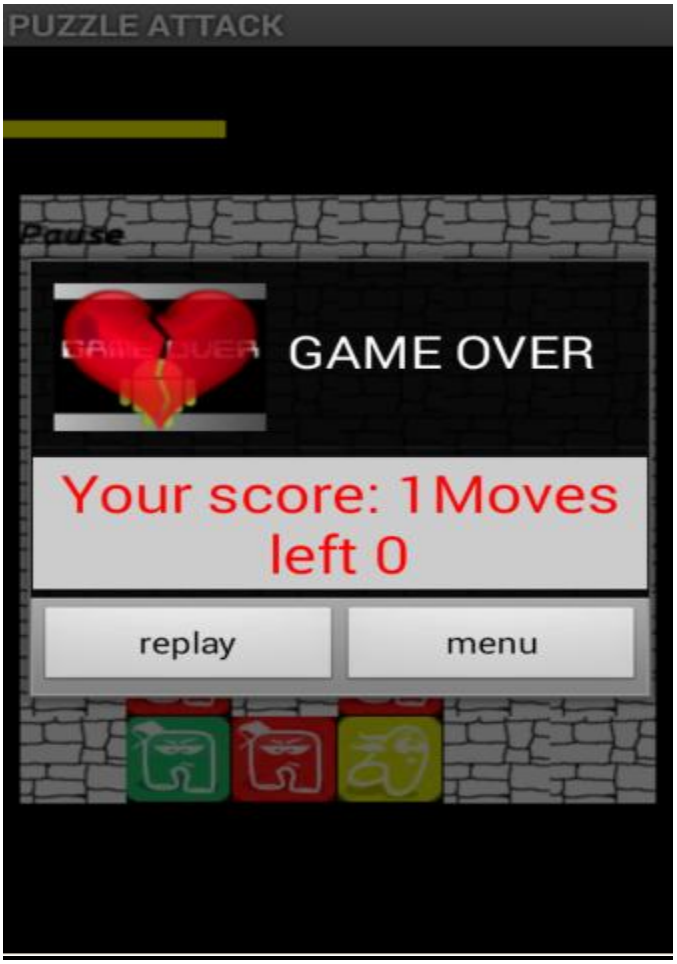


Si le jeu est gagné on a l'affichage suivant avec le score et le bouton Next Level :



Pour commencer chaque stage on a un message comme ceci pour indiquer le numéro de stage :

Un message d'alerte qui indique que la partie est perdu une fois le nombre de permutation épuisé. Le nombre de permutation autorisé est de 3 mouvements.



Enfin nous avons le message pause qui s'active à chaque fois qu'on clique sur le bouton pause qui se trouve tout en haut à droite de notre matrice qui permet d'arrêter et de reprendre le jeu ou de retourner vers le menu principale.

3)3-1 Descriptions des classes et méthodes utilisées :

1) Le code suivant est appelé onTouchEvent :

Ca veut dire quand on touche l'écran cette fonction sert à gérer les interactions tactiles du jeu :

```
public boolean onTouchEvent(MotionEvent event)
{
   int posX = (int) (event.getX() / img[0].getWidth());
   int posY = (int) ((event.getY() - mapTopAnchor) / img[0].getHeight());
   if ((posY >= mapHeight || (event.getY() - mapTopAnchor) <= 0) || (posX >=
mapWidth || (event.getX() - mapLeftAnchor) <= 0)) {
return super.onTouchEvent(event);
   }
 occ++;
```

```
   if (occ > 2)
      occ = 1;
if (occ == 1 && posX !=0 && posY!=0)
   {
      oldI = posY;
      oldJ = posX;
      image2 = map[oldI][oldJ];

   }

   if (occ == 2 && posX !=0 && posY!=0)
   {
      image = map[posY][posX];
      map[oldI][oldJ] = image;
      map[posY][posX] = image2;

   }

   if (!clic)
   {
      if( posX !=0 && posY !=0)
      {
         image = map[posY][posX];
         clic = true;

            if (oldI != posY)
               oldI = posY;
            if (oldJ != posX)
               oldJ = posX;
            image2 = map[oldI][oldJ];

      }
   if(clic)
      numberOfTry--;
   }
   else
   {
      sleep();
 }
```

2) La fonction suivante Vérifie si toutes les cases sont vides et si le joueur gagne ou perd :

```
savescore();

siToutesLesCasesSontVides();
if( posX==0 && posY==0) {

   pauseAlerte();

      posX=oldJ;
      posY=oldI;
}
numberOfTry--;
if (numberOfTry<=0)
   lost();

return super.onTouchEvent(event);
```

3) Apres on le code qui Vérifie si le jeu est terminé et si le joueur gagne ou perd :

```
public void siToutesLesCasesSontVides() {

   int estVide = 0;
   for (int i = 0; i < mapHeight; i++) {
      for (int j = 0; j < mapWidth; j++)

         if (map[i][j] == 0 || map[i][j] == 7 )
            estVide++;

   }
   if (estVide == mapHeight * mapWidth) {
      isWin = 1;
      niveauSuivant++;
      if (isWin == 1 && time > 0)

         won();
      else {
         isWin = 0;
         lost();
      }
   } else
      isWin = 0;


}
```

4) Fonction qui supprime Supprimer les cases vides verticalement :

```
public void supprimerCasesSiPlus3vertical() {


   for (int i = 1; i < mapHeight - 2; i++)

   {
      for (int j = 0; j < mapWidth; j++) {

         if (map[i][j] != 0 && map[i][j] == map[i + 1][j] && map[i + 1][j] ==
map[i + 2][j]) {
            map[i][j] = 0;
            map[i + 1][j] = 0;
            map[i + 2][j] = 0;
            sleep();
         }
      }

   }
   successfull = +3;

}
```

5) Fonction Supprimer les vides horizontalement :

```
public void suppLine(int i, int j, int nb) {


   for (; nb < 0; nb--) {
      map[i][j--] = 0;
      successfull = +1;
   }

   decalerVersBasSiVide();
}
```

6) Fonction qui sert à Déplacer les cases vers le bas si elles se trouvent sur un vide :

```
   public void decalerVersBasSiVide() {

      // Décalage vers le bas si y'a un vide
      for (int i = 1; i < mapHeight; i++)
      {
         for (int j = 1; j < mapWidth; j++)
         {

            if (map[i][j] == 0 && map[i - 1][j] != 0 ) {
               for (int c = i; c > 0; c--) {
                  map[c][j] = map[c - 1][j];
                  map[c - 1][j] = 0;

               }
               sleep();
            }

         }
         j=0; }}
```

7) La fonction pause :

```
   public void sleep() {
      try {
         Thread.sleep(200);
      } catch (Exception e) {
         e.printStackTrace();
```

4) Difficultés rencontrées :

Au cours du développement, nous avons rencontré plusieurs difficultés, comme :

- Bug suite à la mise à jour d'android studio : après mise à jour et passage de la version 1.4 à la version 1 .5, un problème de détection d'adb est survenu, après plusieurs heures de tentative de réparation, on a été obligé de contourner le bug en passant par la méthode manuelle en utilisant les lignes de commandes ;
- La programmation de la fonction pause.

Figure: A propos

5)-Conclusion

Au courant de la réalisation de ce projet nous avons eu l'opportunité de découvrir une nouvelle façon de programmer sur une nouvelle plateforme de travail.

Nous nous sommes familiarisés avec cet environnement pendant la période de la conception et de la réalisation de cette application et nous espérons que ça nous ouvrira de nouvelles portes.