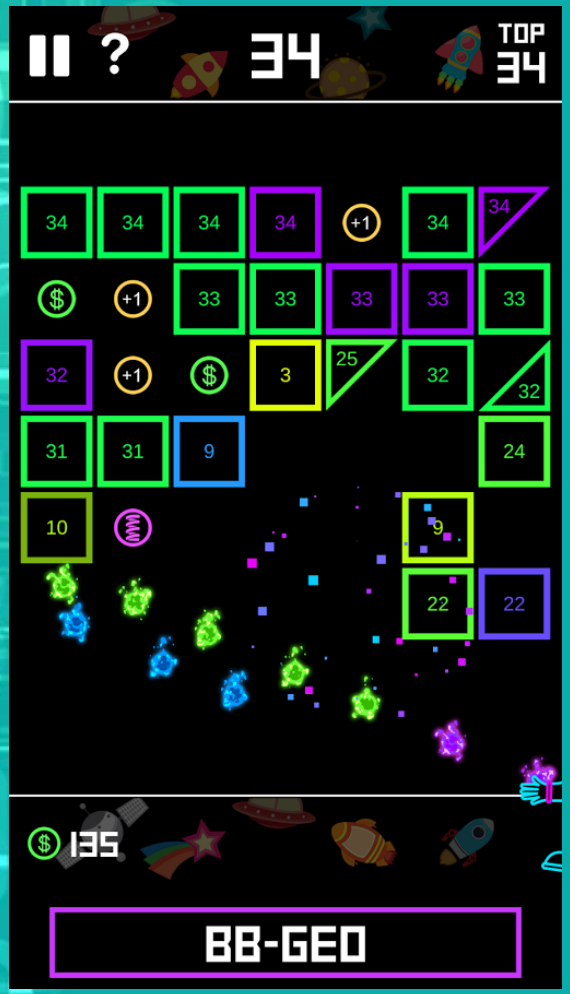
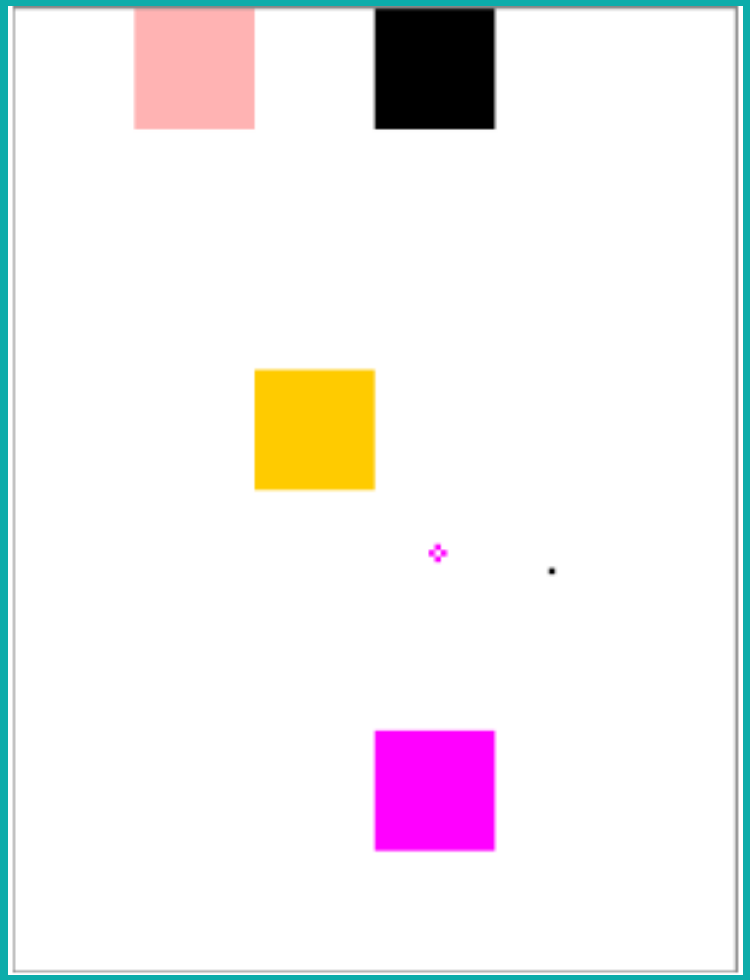
Jeu de casse brique

MESNIER Adrien
Avec BRETON Gwendal

Introduction



Le jeu BBGEO dont nous nous sommes inspirés



Notre programme

Plan

I/ Le fonctionnement du programme

1) Fonctionnement global

2) La procédure *lancer_balle*

3) La fonction *obstacle_present*

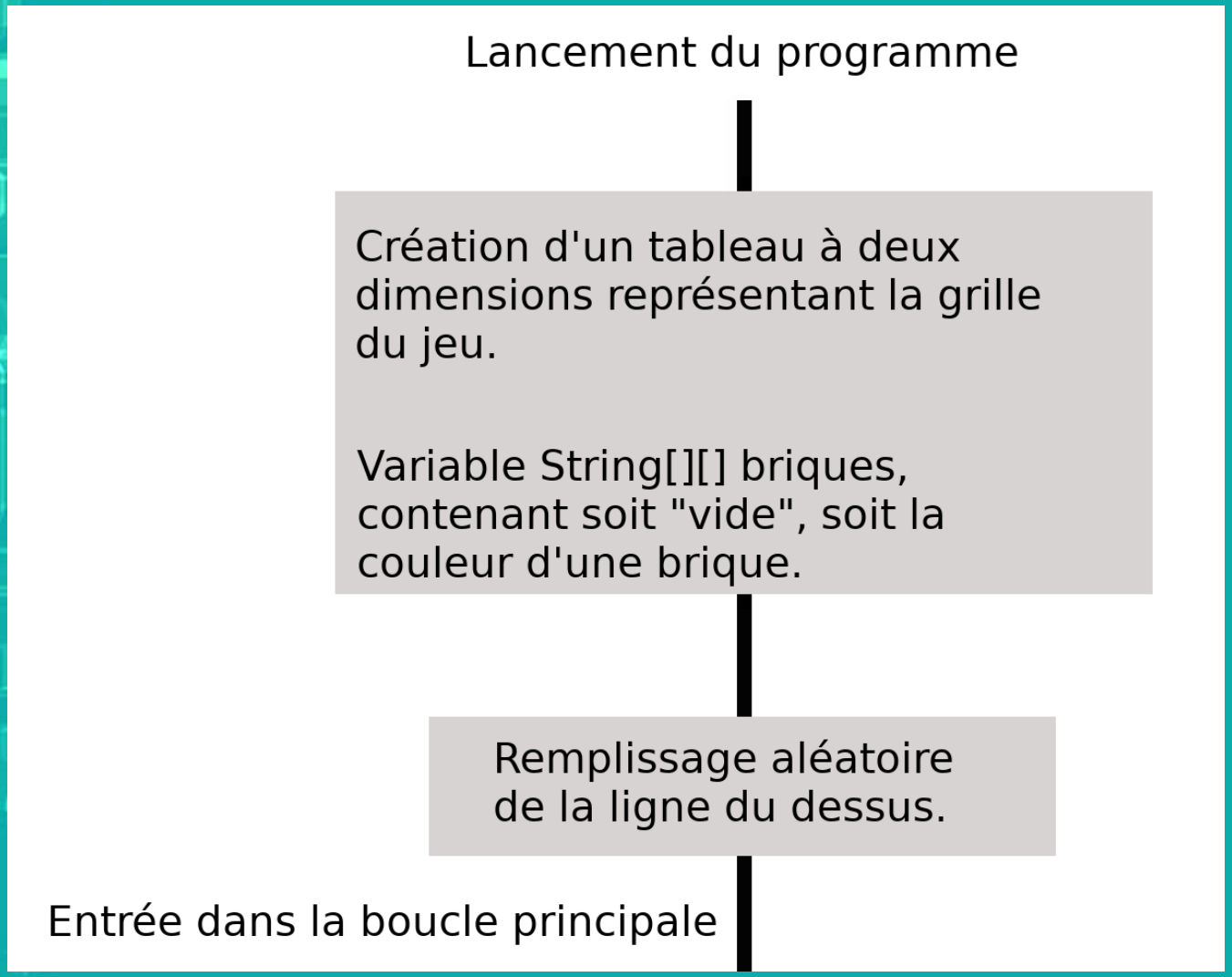
4) La fonction *finDeTour*

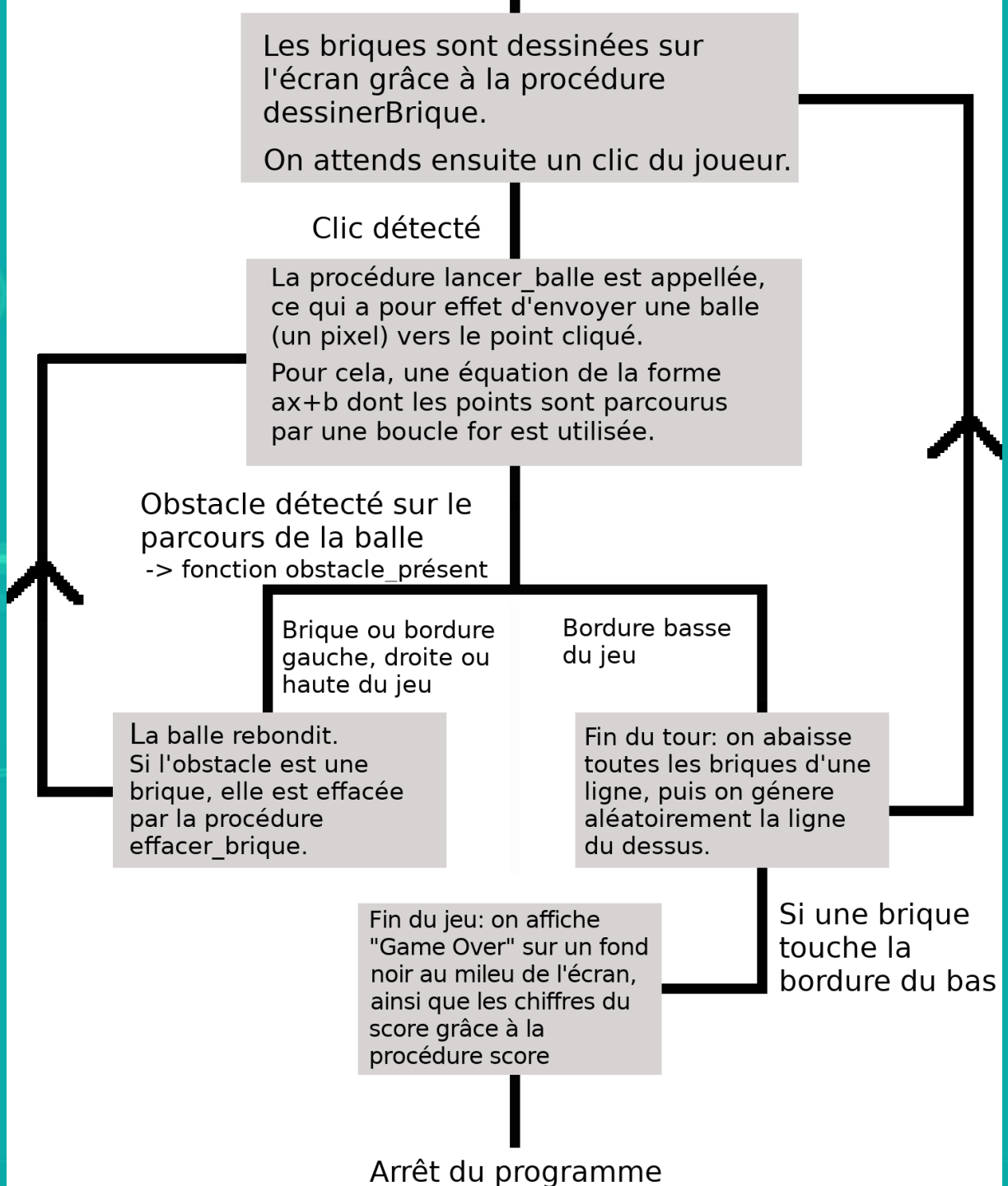
II) Bilan

1) Sur le programme

2) Sur la réalisation du projet

Fonctionnement global




Les briques sont dessinées sur l'écran grâce à la procédure dessinerBrique.
On attends ensuite un clic du joueur.
Clic détecté
La procédure lancer_balle est appellée, ce qui a pour effet d'envoyer une balle (un pixel) vers le point cliqué.
Pour cela, une équation de la forme ax+b dont les points sont parcourus par une boucle for est utilisée.
Obstacle détecté sur le parcours de la balle
-> fonction obstacle_présent
Brique ou bordure gauche, droite ou haute du jeu
Bordure basse du jeu
La balle rebondit.
Si l'obstacle est une brique, elle est effacée par la procédure effacer_brique.
Fin du tour: on abaisse toutes les briques d'une ligne, puis on génere aléatoirement la ligne du dessus.
Si une brique touche la bordure du bas
Fin du jeu: on affiche "Game Over" sur un fond noir au mileu de l'écran, ainsi que les chiffres du score grâce à la procédure score
Arrêt du programme

La procédure lancer_balle

```
if(a == null || direction == null){
      //On calcule l'equation de la droite (on convertit en float pour plus de prÃ©cision)
      a = ( (float) yArrivee-(float) yDepart) / ((float) xArrivee - (float)xDepart);
      //si la balle part vers la gauche
     if (xArrivee <= xDepart) direction = -1;
     //Ou vers la droite
     else if (xArrivee > xDepart) direction = 1;
}
```

La procédure lancer_balle

```
//Si la balle rencontre une brique ou un rebord, on la fait rebondir
if(obstacle == "bordure_basse") {
     break;
}
else if(obstacle.indexOf("coin") != -1){
     lancer_balle(x, y, null, null, a, -direction);
}
else if (obstacle.indexOf("haut") != -1 || obstacle.indexOf("bas") != -1){
     lancer_balle(x, y, null, null, -a, direction);
}
else if (obstacle.indexOf("droit") != -1 || obstacle.indexOf("gauche") != -1){
     lancer_balle(x, y, null, null, -a, -direction);
}
break;
```

La fonction obstacle_present

```
String obstacle_present(int x, int y) {
   //on regarde si la balle a atteint une bordure
     if (y <= 0) return "bordure_haute";
     else if (y >= hauteur * tailleBrique) return "bordure_basse";
     else if (x <= 0) return "bordure_gauche";
     else if (x >= largeur * tailleBrique) return "bordure_droite";
     else {
          //La balle n'a atteint aucune bordure, on regarde maintenant si
          //On cherche quelle ligne et colonne du tableau de briques corr
          int xTab = (int)floor(x / tailleBrique);
          int yTab = (int)floor(y / tailleBrique);
          if (briques[yTab][xTab] == "vide") return "aucun";
          else {
               //La balle a atteint une brique, il faut maintenant savoir
               int yHaut = yTab * tailleBrique,
               yBas = yTab * tailleBrique+tailleBrique-1,
               xGauche = xTab * tailleBrique,
               xDroite = xTab * tailleBrique+tailleBrique-1;
               //DÃ©jÃ  on vÃ©rifie si c'est arrivÃ© sur un coin
               if (x == xGauche && y == yHaut) return "brique_coin";
               else if (x == xDroite && y == yHaut) return "brique_coin";
               else if (x == xGauche && y == yBas) return "brique_coin";
               else if (x == xDroite && y == yBas) return "brique_coin";
               //Sinon, on cherche le cÃ´tÃ©
               else {
                    if (y == yHaut) return "brique_haut";
                    else if (y == yBas) return "brique_bas";
                    else if (x == xGauche) return "brique_gauche";
                    else if (x == xDroite) return "brique_droit";
                    //Au cas oÃ¹
                    else return "brique";
          }
     }
   }
}
```

La fonction finDeTour

```
90  boolean finDeTour() {
91      //On abaisse toutes les briques d'une ligne, et on genere la ligne du dessus
92      //Si la ligne du bas n'est pas vide avant d'Ãªtre abaissÃ©e, la partie s'arrÃªte
93      for (int i = briques.length -1; i >= 0; i --) {
94          for (int j = 0; j < briques[0].length; j ++) {
95              //Si une brique dans la ligne du bas n'est pas vide
96              if (i == briques.length -1 && briques[i][j] != "vide") {
97                  return  false;
98              }
99              //Si on est pas sur la ligne du dessus, on copie la brique au dessus
100             if (i != 0) {
101                 briques[i][j] = briques[i -1][j];
102             } else {
103                 //si on est sur la ligne du dessus, on genere la brique au hasard
104                 briques[i][j] = brique_au_hasard();
105             }
106         }
107     }
108     return  true;
109 }
```

Bilan

- Le programme peut encore être amélioré
- La réalisation du projet a été enrichissante