

---

# Rapport de projet

---

Feruglio Hugo - Barbosa Juliana - Boukhizzou Houssam  
Truong Pauline - Benabadji Waël - Hubert Thibault

---



---

# Sommaire

*I. Le jeu*

*II. Les problèmes rencontrés*

*III. Notre organisation*

*IV. Les fonctions*

*V. Les IA*

*VI. Conclusion*

---

## *Le jeu*

Bombe Party : un jeu a plusieurs objectifs :

Pour commencer, nous voulions créer un jeu vraiment simple à comprendre mais avec une certaine réflexion sur les stratégies.

L'arène est un tableau, une sorte de grille dans laquelle chacune des cases contient un chiffre aléatoire entre 0 et 99.

Votre but sera alors de vous placer dans les cases comportant le plus grand chiffre. Ces derniers s'additionneront à votre score sauf qu'à chaque fois que vous bougerez, vous placerez une bombe sur là où vous étiez.

Pour compliquer la chose, chacune des bombes explosera 3 tour après et dans un périmètre d'une case autour : les 8 cases.

L'objectif sera donc double : gagner le plus de points et éviter de mourir à cause des bombes.

## *Son fonctionnement*

Les IA devront parcourir le tableau chacune à leur tour pour se placer dans les cases qu'elles veulent.

En arrivant sur une case, elles prendront son score puis la valeur de « 0 » lui sera affecté, ce qui signifie que la bombe est enclenché

---

Les 2 objectifs étant de rester en vie et de gagner le plus de points diversifie la façon de gagner la partie. En effet, si une IA va sur une bombe, elle meurt et l'autre gagne. Mais dans le cas où toutes les IA ont esquivés les bombes et que tout le tableau est rempli, alors le jeu s'arrête et l'IA ayant le meilleur score gagnera la partie.

### *Les problèmes rencontrés*

Ils ont été nombreux, et pourtant il fallait les résoudre rapidement.

Comme beaucoup d'élèves, nous avons commencé le projet avec la SDL en rêvant de faire un jeu super design mais cela était plus compliqué que prévu.

L'installation de la SDL nous avait pris de nombreux jours, et étant un groupe travaillant sur Mac et Windows cela ne nous arranger en rien.

Après avoir passé des heures, gâcher des séances projet à l'école seulement pour avancer via la SDL, c'est au moment d'installer la librairie « ttf » (qui nous aurait permis d'insérer : Chaines de caractères dans notre tableau) que nous nous sommes rendu compte que nous perdions trop de temps.

Le temps était précieux, et il fallait vite passer à l'action. C'est alors que nous avions décidé de faire le projet sans SDL, tout en essayant d'avoir un jeu qui marche, puis améliorer le design à la fin s'il nous restait du temps.

## Nos méthodes de travail

Travailler en groupe était parfois compliqué car nous avons eu des problèmes de communications lors des premiers moments de travail.

Afin de ne laisser seul personne d'entre nous sur une tache qu'il n'arriverait peut être pas à effectuer nous avions opté pour choix de travailler par 2 : il y avait donc 3 groupes.

Toute la communication s'était faite via un groupe Facebook où tout le monde était impliqué dans l'avancement du projet.



---

## *Les fonctions principales*

Notre code n'est pas énorme, il fait environ 250 lignes et ne contient pas énormément de fonctions, alors présentons les principales d'entre elles :

**\*\*create\_map()** : Cette fonction permet l'allocation dynamique de la mémoire, qui était une condition essentiel a intégré dans l'arène étant donné que nous avions travaillé dessus.

**\*\*fill\_map()** : Cette fonction va permettre l'ajout d'un chiffre aléatoire dans notre arène. Nous utilisons le macro RANDOM que nous avons initialisés à 100 et qui est changeable facilement.

**free\_map()** : Cette fonction permettra de libérer la mémoire que Nous avons alloué dans la fonction **\*\*create\_map()**

**display\_map()** : Cette fonction permet la création de notre tableau. Elle ressemble à nos fonctions du projet  
COLORWARS

**check\_end\_game()** : Cette fonction, comme son nom l'indique permet de vérifier quand la partie est finit.

Nous parcourrons tous le tableau, en additionnant toutes les cases et si c'est égale à 0 alors dans ce cas-là toutes les cases ont été pris et le jeu est finit.

**bomb()** : Cette fonction permet la gestion des bombes ainsi que

---

son périmètre.

## *Les IA*

Lors de ce projet, et pour montrer comment marcher notre jeu, nous avions créé 2 IA : une intelligente et une normal.

L'IA random se plaçait aléatoirement dans le tableau, et parfois Elle restait même à la case où elle était au tour précédent.

L'IA intelligente parcourt tout le tableau, et après avoir vu toutes les cases, elle se place sur le plus gros chiffre.

Ce sont 2 principes différents mais il serait intéressant de voir d'autre IA avec des fonctionnements différents...

## *Les améliorations*

Nous aimons notre jeu mais il manque vraiment de design. Une mise en place sous SDL, un ajout de couleur, beaucoup d'idée sont à mettre en place afin que notre jeu soit plus attrayant.

## *Conclusion*

Pour finir, cette expérience a été enrichissante pour chacun de nous, puis les travaux de groupe nous permettent de mieux communiquer, ce qui est indispensable pour un bon ingénieur.

---

Le travail sur les IA arrive et nous sommes pressés de devoir élaborer des stratégies pour que le jeu prenne vraiment sa forme amusante.

### Annexe :

Voici une capture de BombeParty : les scores des IA sont sous le tableau et sont actualisés à chaque tour. (1ère photo)

Lorsque le jeu se finit, tout s'arrête et le nom du gagnant apparaît.

89	67	89	84	00	00	46	82	00	19
73	30	43	86	86	33	11	25	18	63
41	54	49	47	66	00	13	54	55	17
66	13	29	00	00	63	00	02	01	06
07	14	45	00	40	00	00	30	25	27
04	43	51	78	00	55	33	00	13	78
29	21	60	43	53	41	00	70	41	00
60	00	00	00	04	80	45	23	81	15
73	22	54	29	27	00	12	84	91	85
02	13	00	49	81	42	00	49	89	12

Scores: j1 = 356 ; j2 = 1083;

Scores: j1 = 493 ; j2 = 1441;  
Player 2 wins with 1441 points, player 1 dead with 493 points