[Date]

CHARDONNET Romain (qualité)

EL MANSOURI Redouane (chef)

HURDEBOURCQ Jeremy

ISAAC Elie

groupe 9

CESU

Serious game

Table des matières

[I) Introduction 2](#_Toc41097823)

[II) Fonctionnalités 2](#_Toc41097824)

[a) Extraction 2](#_Toc41097825)

[1) Présentation 2](#_Toc41097826)

[2) Méthode 5](#_Toc41097827)

[b) Triage 6](#_Toc41097828)

[c) Evacuation 7](#_Toc41097829)

[1) Présentation 7](#_Toc41097830)

[2) Les fonctionnalités détaillées selon le code 9](#_Toc41097831)

[d) Centralisation des informations 10](#_Toc41097832)

[1) Présentation 10](#_Toc41097833)

[2) Les fonctionnalités détaillées selon le code 12](#_Toc41097834)

[III) Outils 12](#_Toc41097835)

[a) MVC 12](#_Toc41097836)

[b) Login 12](#_Toc41097837)

[c) Chat 12](#_Toc41097838)

[d) Diagramme de classes 12](#_Toc41097839)

[e) Base de données 12](#_Toc41097840)

[IV) Gestion de projet 12](#_Toc41097841)

[V) Conclusion 12](#_Toc41097842)

# Introduction

Le projet CESU est un serious game de simulation d’urgence dans le but de contribuer aux formations du CHU de Dijon. Le jeu permet donc la connexion de plusieurs joueurs à l’aide d’un login et de communiquer à l’aide d’un chat afin de simuler le fonctionnement d’une situation d’urgence en équipe. L’application est constituée de trois grandes parties principales :

* L’extraction des victimes : c’est dans cette partie que le jeu commence, le joueur place les véhicules d’urgence et procède à l’extraction des victimes par les pompiers pour les déposer dans le PMA (poste médical avancé).
* Triage : il a lieu au PMA, les médecins du Samu procèdent au triage des victimes en fonction de leur cas respectif.
* Evacuation : cette partie est liée à l’évacuation des victimes dans les ambulances, on y trouve les ressources disponibles (nombre de places,…).

Le projet a été développé sous XAMPP pour la simulation en local lors du développement personnel. Ce dernier a été judicieux puisqu’en plus de faire fonctionner Apache, il intègre PhpMyAdmin. Nous avons donc pu aisément travailler avec une base de données local. Le projet dans sa globalité est hébergé sous le serveur Aragon de l’IEM.

Le projet a été développé avec les outils suivants : HTML, CSS, JQuery, Php et SQL (pour les requêtes gérées avec Php).

# Fonctionnalités

## Extraction

### Présentation

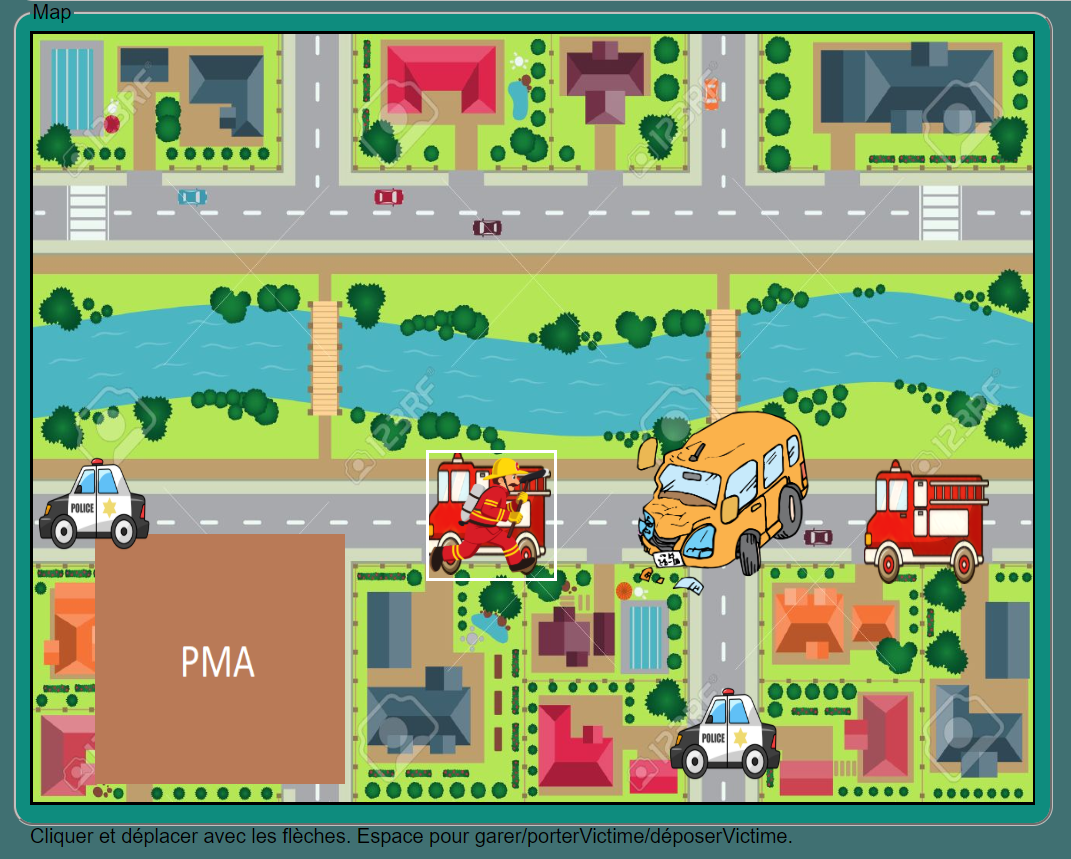
La « première » partie de l’application est l’extraction. Elle possède un genre de « gameplay séquentiel » en quelque sorte puisque plusieurs étapes se succèdent dans cette partie. Lorsque l’on arrive sur cette page, on remarque une carte représentant une ville avec un incendie.

La première étape consiste à placer les véhicules d’urgence de manière stratégique aux alentours de l’incident. Les instructions sont données, un clic suivi des touches fléchées pour déplacer les véhicules puis espace pour immobiliser le véhicule. En tout, il y a deux véhicules de police et deux véhicules des pompiers à garer. On suppose pour cette partie que le Samu est déjà sur place et a installé le PMA. Celui-ci est donc fixé dès le début.

Une image contenant capture d’écran

Description générée automatiquement

Lors de l’immobilisation d’un véhicule des pompiers, son pompier respectif en sort pour procéder à l’évacuation des victimes.



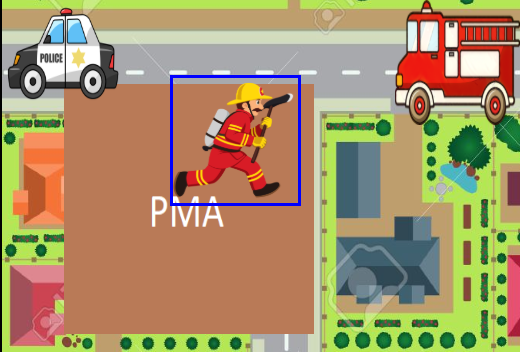
Ensuite, les pompiers doivent procéder à l’extraction des victimes. Lorsqu’un pompier rentre dans la zone de danger, son contour devient jaune, on peut donc extraire une victime.



Lorsque le pompier a récupéré une victime, son contour devient rouge, et le nombre de victimes est décrémenté au-dessus.



On déplace le pompier dans la zone du PMA. Son contour devient bleu lorsque l’on peut déposer la victime.



Lors de l’appui sur « espace » à nouveau, le contour redevient blanc, et la victime à bien été déposée dans le PMA. C’est-à-dire qu’on a fait une insertion dans la base de données. Pour ce faire, on utilise un appel Ajax dans le fichier .js correspondant.

### Méthode

Le fichier vue1.js traite les fonctionnalités suivantes :

* Utilise les instance javascript des personnages et véhicules concernés
* Sélection des éléments cliqués en mettant le contour en blanc
* Déplacement de l’élément sélectionné avec les touches fléchées à l’aide d’une fonction move
* Adaptation de la couleur du contour à l’aide d’une fonction checkZone
* Garer un véhicule avec une méthode garer
* Porter une victime à l’aide d’un méthode porteVictime
* Déposer une victime dans le PMA à l’aide d’une méthode deposerVictime
* Génération de noms et nombres aléatoires à l’aide de deux fonctions respectives

Nous détaillerons par la suite la partie de dépôt d’une victime dans le PMA puisque c’est la partie qui va interagir avec le reste du projet. Cette fonction est appelée à chaque dépôt de victimes. On y trouve à l’intérieur un appel ajax vers le fichier qui appelle le contrôleur correspondant.

On utilise ici le type « post » puisque c’est la convention lors d’un enregistrement dans le serveur. Dans le cas d’une récupération de données, on aurait utilisé « get » mais dans les deux cas le système fonctionne.

Les données qu’on envoie dans « data » correspondent à chaque colonne de la table Victime présente dans la base de données. Pour les noms et les nombres, on utilise des fonctions de génération aléatoires.

Le nom de la fonction « insert » est également renseigné lors de l’envoi des données pour que le fichier cible sache quelle fonction du contrôleur appeler. Le reste des données envoyées sont :

* Le nom qui est une chaine de caractères générée aléatoirement
* Le prénom qui est également une chaine de caractères générée aléatoirement
* Un booléen « vivant » pour indiquer si la victime est en vie
* Un entier « vie » indiquant la jauge de vie d’une victime
* L’état de la victime également représenté par un nombre entier :
  + 0 si son état est normal
  + 1 en cas de blessure légère
  + 2 en cas de blessure sévère
  + 3 en cas de blessure grave
  + 4 en cas de décès
  + 5 en cas de blessure psychologique

Le contrôleur récupère ensuite les données à traiter en paramètres de la fonction « insertVictime ». Une manœuvre est faite dans cette fonction pour ajuster des variables en fonction des autres pour garder une cohérence lors avant leur stockage dans la base de données par le modèle.

Pour ce faire, on procède en deux étapes. La première consiste à vérifier la variable « $vivant », si elle vaut zéro, on met « $vie » également à zéro et « $etat » à 4.

La deuxième étape consiste à ajuster la vie en fonction de l’état de la victime dans le cas où elle n’est pas morte. On assigne à la vie un nombre aléatoire appartenant à une plage conforme avec l’état de la victime. Enfin, le modèle renvoie les données ainsi ajustées.

## Triage

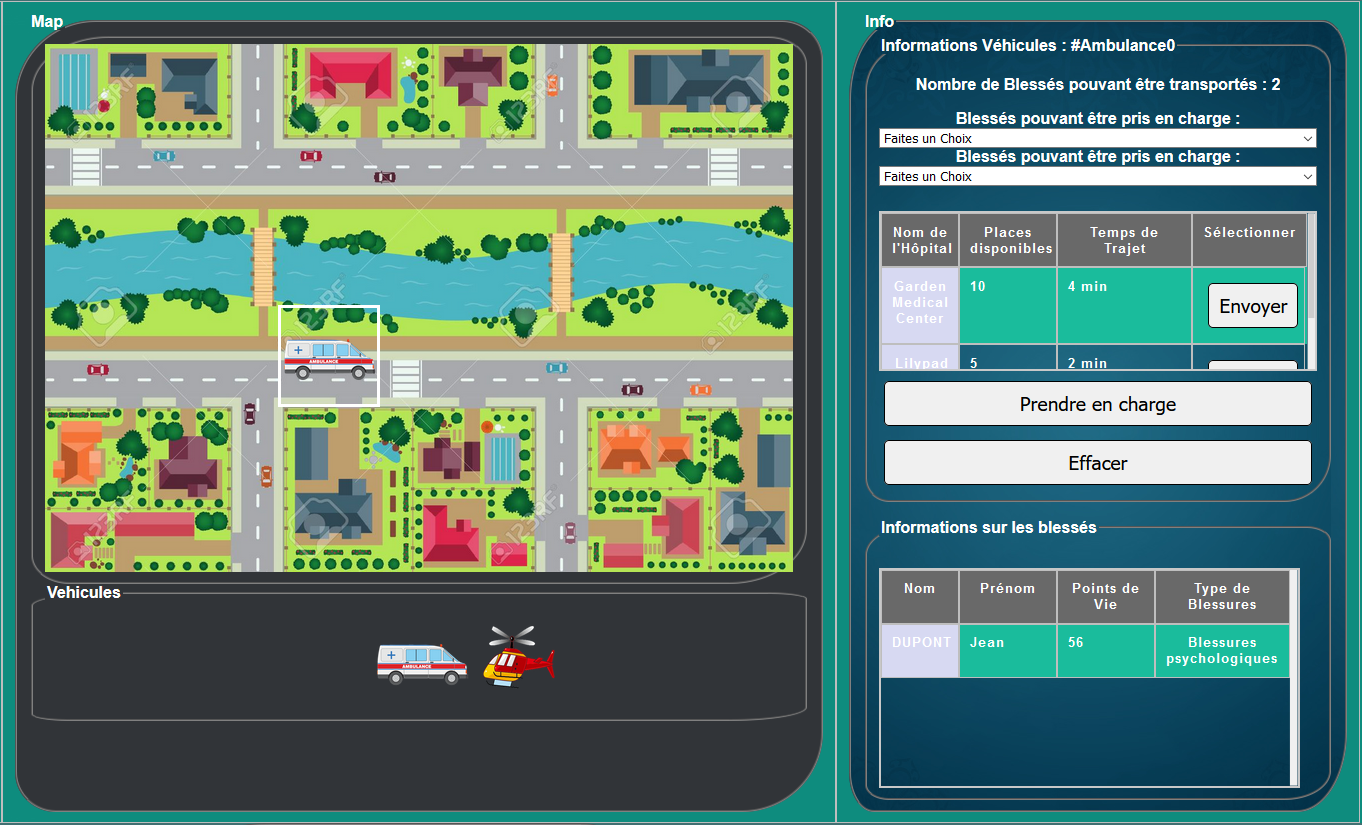
## Evacuation

### Présentation

Observons maintenant la « partie » évacuation des blessés traités par la « partie » Triage de cette application. Elle est composée d’une vue de la ville très limitée, d’une liste de véhicules mis as la disposition ainsi que d’un tableau de blessés envoyés par la PMA.

Dans un premier temps, le joueur étant affecté à cette partie doit placer les véhicules qui lui sont mis a ça disposions sur l’image représentant la ville. Pour chaque sélection de véhicules, les informations de ceci sont affichées dans un encart "Informations véhicules". De plus, le véhicule sélectionné est présentement détaillé par un cadre blanc autour de son image.

Lors de la sélection d’un véhicule, l’on peut observer dans l’encart « Informations véhicules » les données nécessaires afin d’envoyer les blessés traités par la PMA dans les hôpitaux les plus proches. Ces hôpitaux sont listés dans un tableau comme l’on peut l’observer sur l’image précédente. Les blessés pris en charge et traité par la PMA sont ensuite listé dans le tableau « Informations sur les blessés » permettant ainsi au joueur d’avoir une vue sur la liste des patients à envoyer aux hôpitaux les plus proches.



Après la sélection d’un véhicule, l’on a la possibilité de choisir à partir de la liste présentes dans « Informations sur les blessés », les blessés qui pourront partir pour l’hôpital choisi au préalable. Chaque type de véhicule possède un nombre de places limitées.

Par exemple, prenons l’« #Ambulance0 ». On a sélectionné un blessé ainsi qu’un hôpital.



Ensuite, l’on sélectionne le bouton « Prendre en charge » qui bloquera ainsi notre véhicule et nous ne pourrons plus l’utiliser selon le temps de trajet de l’hôpital choisi (temps divisé par trois en secondes pour des raisons de gameplay). Le blocage du véhicule est animé selon le fait que le joueur s’il veut déplacer ce même véhicule, il ne le pourra et son contour sera affiché en rouge. De plus, un récapitulatif des informations de ce même véhicule lors de sa sélection sera disponible dans l’encart « Informations véhicules » à la place de l’affichage précédant. 

L’on peut observer de même, que la liste des blessés lors de la « Prise en charge » par un véhicule, est mise à jour. Et ainsi de suite.

### Les fonctionnalités détaillées selon le code

Les fichiers présentant ces diverses images vues précédemment dans cette partie et permettant cette simulation commencent tous par le préfixe « EVAC ». Le fichier principal traite donc les fonctionnalités suivantes :

• Utilise les instance JavaScript des personnages et véhicules concernés.

• Sélection des éléments cliqués en mettant le contour en blanc en JavaScript

• Déplacement de l’élément sélectionné à l’aide de la souris grâce à l’élément « draggable » de JQUERY

• Modification de la couleur du contour d’un élément à l’aide d’une variable de ce même élément en javascript

• Récupération des blessés depuis une base de données remplie par l’envoi de la PMA en traitant des informations depuis des fichiers PHP remontant à un script

• Des animations d’informations depuis les divers scripts « EVAC ».

Pour des raisons de lisibilités, je vous renvoie aux codes commentés et expliqué de chacun des fichiers issus de cette partie.

## Centralisation des informations

### Présentation

Dans cette partie, le joueur affecté à ce rôle, aura donc pour lui une vue globale des actions de chacun des joueurs, mais il ne pourra pas directement interagir sur chacun des éléments présents sur la carte détaillée.



L’on peut cliquer sur chacun des éléments afin d’avoir des informations les concernant. Celles-ci sont visibles par-dessus cette page à partir d’un overlay.



Pour fermer cet overlay, il suffit de cliquer dessus et ainsi de suite. Comme le nom de cette partie l’indique, toutes les informations concernant la liste des blessés envoyés au PMA, la liste des blessés pris en charge par l’évacuation ainsi que la liste des hôpitaux disponibles sont regroupées dans l’encart « Info » de droite de la page.



Les informations des éléments et des listes sont régulièrement mises à jour toutes les 5, voir toutes les 30 secondes depuis le début de la simulation.

Cette centralisation des diverses informations de chacune des parties aurait dû permettre au joueur possédant cette partie d’interagir avec chaque joueur afin d’envoyer si ceux-ci le désiraient des renforts par le biais du tchat. Or, par soucis de temps et de notre groupe, cette fonctionnalité n’a pas pu voir le jour.

### Les fonctionnalités détaillées selon le code

Comme pour l’évacuation, les fichiers permettant la centralisation des informations commencent par le préfixe CRRA. Le fichier principal implémente donc les fonctionnalités suivantes :

* Des animations issues des scriptes permettant entre autres le placement des divers éléments, les informations de ceux-ci et le remplissage des différentes listes.
* ….

Pour parfaire le placement de chacun des éléments présent sur cette centralisation des données, nous avons implémenté une fonction sendData() dans les scriptes des parties « Evacuation », « Extraction » et « PMA » afin de stocker chacune des informations dans des fichiers JSON que l’on a ensuite extrait à partir des fichiers PHP et scriptes « CRRA ».

# Outils

## MVC

## Login

## Chat

## Diagramme de classes

## Base de données

# Gestion de projet

# Conclusion