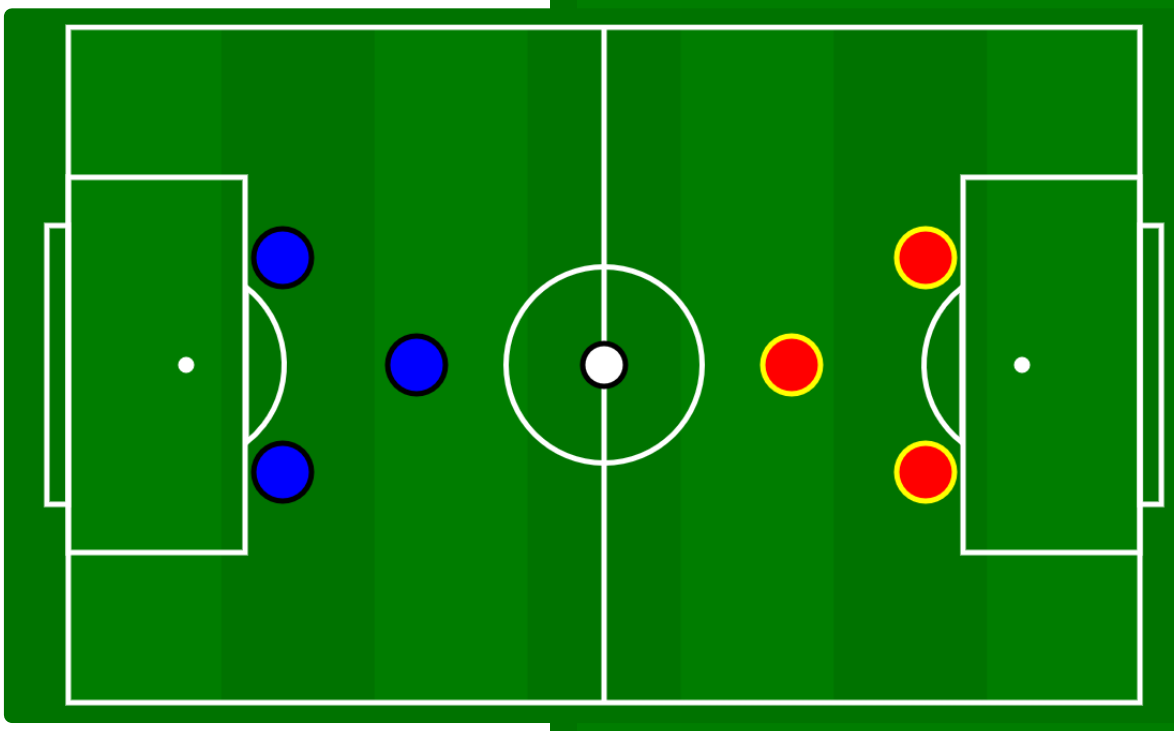
Création d'un billard original version football en JavaScript



Site : http://oc3.lddr.ch/Sacha/

Fichiers : https://github.com/TinguelySacha/tm

Lycée Denis-de-Rougemont

3ème année

Sacha Tinguely 3MG04

Christophe Bruchez

TABLE DES MATIERES

INTRODUCTION

Pour introduire le sujet de la création de ce jeu en ligne, il paraît nécessaire de commencer par expliquer comment l'idée de ce jeu est apparue.

Pour la première partie du jeu, la partie *pendu*, il s'avère que la conception de cette dernière remonte à la fin de la première année de lycée. Ayant appris quelques bases de Python, je voulus tenter de mettre en place un petit projet à la portée de mes connaissances de l'époque. Ainsi, je me suis dit que la programmation d'un pendu serait atteignable et je me mis à sa création. Quelques heures plus tard, le programme était fini et sauvegardé au fin fond de mon ordinateur, pensant ne jamais en avoir besoin.

Cependant, quelque temps plus tard, ayant acquis des connaissances naissantes en programmation web, je voulus à nouveau les mettre en œuvre pour un projet quelconque. Je repris donc le programme que j'avais fait en Python et le mis sous forme de site web. C'est de là que viennent les bases de la partie *pendu*.

Après avoir appris que mon travail de maturité consisterait en la création d'une application web, je me suis dit que je pourrais reprendre les bases de ce pendu et les développer afin d'en faire quelque chose de plus intéressant. C'est ici que me vint l'idée de la deuxième partie de ce jeu.

Concernant cette dernière, il se trouve que dans mon enfance, il m'arrivait de jouer à un jeu nommé *Plato*. Ce dernier n'est pas réellement un jeu en lui-même, mais plutôt une plateforme qui permet à ses utilisateurs de jouer à divers jeux entre amis (Echecs, Monopoly, UNO, etc.). Parmi la plénitude de jeux proposés, mes amis et moi jouions la plupart du temps au *Table Soccer* car il était amusant et permettait de jouer à quatre. C'est de là qu'est venue l'idée de la deuxième manche que je vais par conséquent nommer la partie *Table Soccer* tout au long de ce rapport.

C'est donc de l'union de ces deux jeux a priori incompatibles qu'est né le jeu que j'ai décidé de nommer *PenduFoot*.

DEROULEMENT ET BUT DE LA PARTIE

Déroulement et but de la partie

Au lancement du site, l'utilisateur arrive sur un menu lui proposant deux modes de jeu : Le mode *deux joueurs*, permettant de jouer à deux utilisateurs sur un ordinateur et le mode *en ligne*, permettant de jouer contre un autre utilisateur à travers le monde.

Quel que soit le choix de l'utilisateur, le concept du jeu reste le même. Il commencera le jeu par la partie *pendu* dans laquelle il devra essayer de deviner le nom de cinq joueurs de football en l'espace de 30 secondes. Pour ce faire, il lui sera donné la nationalité, le club et le nombre de lettres contenues dans le nom du joueur. Le nombre de joueurs que l'utilisateur réussira à trouver déterminera le nombre de pions qu'il aura au tour suivant, il est donc capital de réussir à deviner le nom d'un maximum de joueurs afin de partir avec un avantage à la manche suivante.

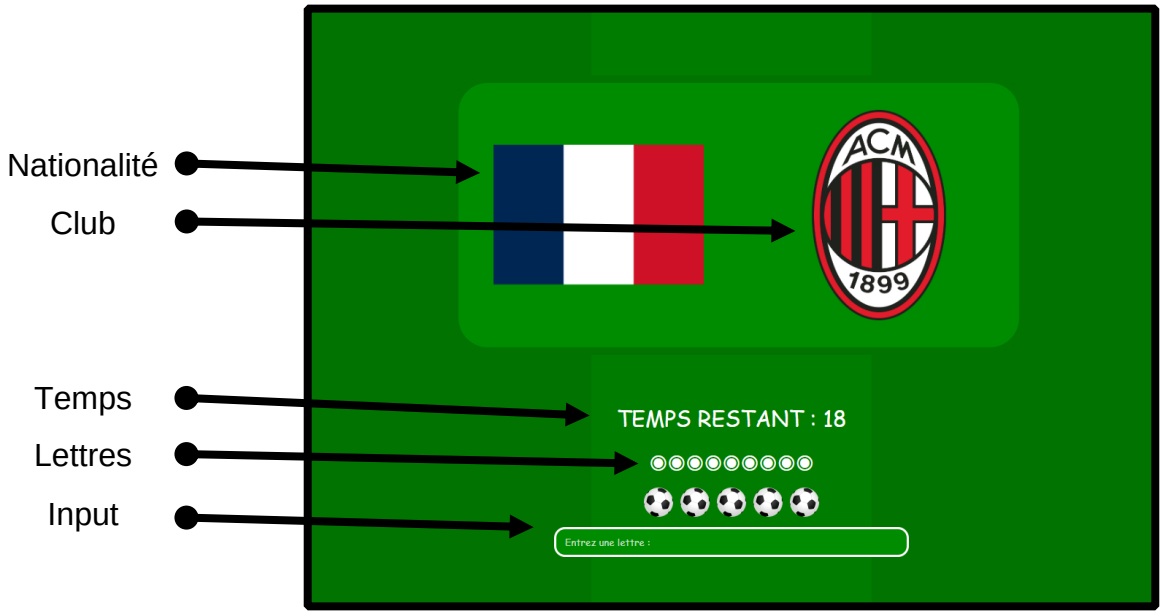


Figure 1: Pendu

La manche suivante justement, consiste en une sorte de billard version football. Chaque utilisateur possède donc un certain nombre de pions en fonction de son score à la manche précédente. Lorsque c'est au tour de l'utilisateur de jouer, ses pions habituellement entourés de noir se retrouvent entourés de jaune. L'utilisateur doit alors sélectionner un pion et le « tirer » dans l'objectif de mettre la balle au fond des cages adverses. La partie prend fin une fois qu'un des deux utilisateurs réussit à marquer trois buts.

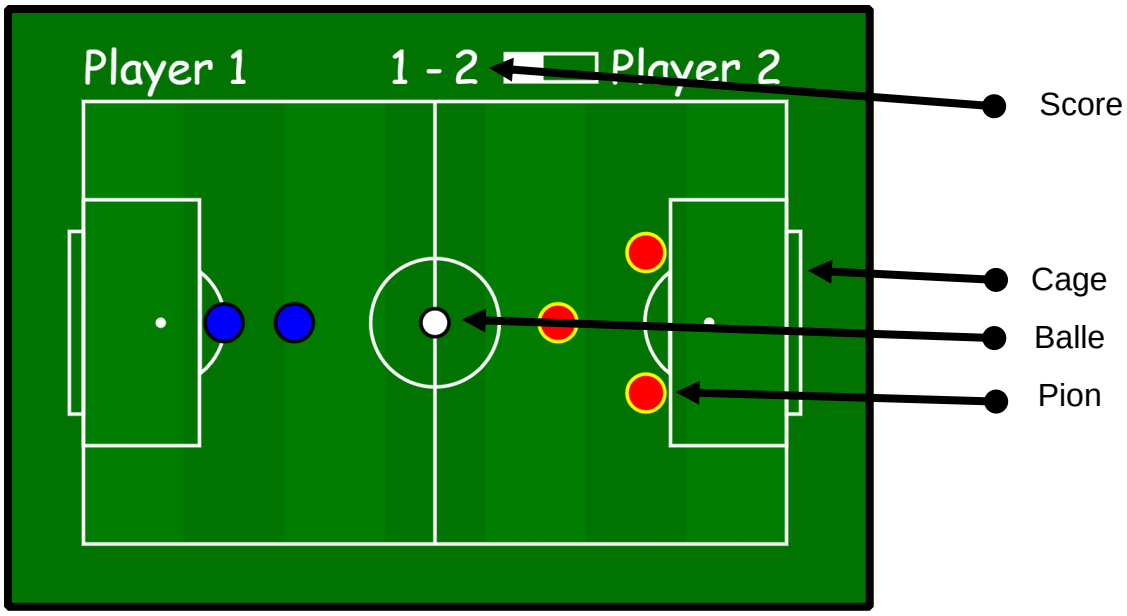
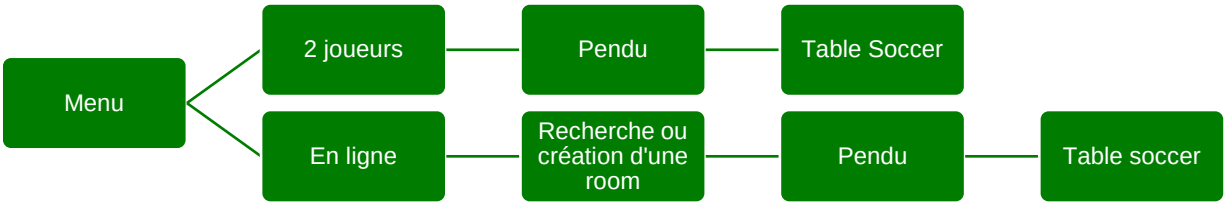


Figure 2 : Table Soccer

Structure du jeu

La structure du jeu avec ses différents modes et son système de manches peut sembler complexe, voici donc un graphique pour en simplifier la compréhension :



FONCTIONNEMENT DU JEU EN LOCAL

Animation CSS3

Dans le but de rendre l'interface du menu plus dynamique, j'ai usé d'animations CSS3. Pour ce faire, il faut utiliser des « keyframes » (images clés). Ces dernières indiquent au programme la position voulue à un certain moment de l'animation. Après avoir mis en place les « keyframes » et avoir donné un nom à l'animation, il suffit d'aller dans le code CSS de l'objet à qui l'on veut attribuer l'animation et de lui spécifier d'effectuer l'animation en question, la durée de celle-ci et le nombre de fois qu'il faut l'effectuer.

Liste des joueurs

Pour que le jeu prenne des joueurs au hasard, il faut commencer par créer une table contenant le nom, la nationalité et le club des joueurs dans une base de données MySQL. Il faut ensuite que le programme la récupère, pour cela, il faut donc quelques lignes de PHP qui accèdent à la base de données, récupèrent la table et la rende accessible à travers tout le programme. Par la suite, il faut que le programme prenne un joueur au hasard et adapte le drapeau, le club, et le « nombre de lettres » à ce dernier. Pour adapter le drapeau et le logo du club, le programme change la source de l'image originale pour la remplacer par celle du drapeau et du logo contenus dans les dossiers « imagescdm » et « imagesldc » sur le serveur.

Pendu

Une fois que tout est mis en place, il faut maintenant faire en sorte que lorsque l'utilisateur rentre une lettre, le programme puisse voir si elle est dans le nom du joueur. Pour cela, le programme commence par prendre le caractère entré dans l'input et le met en minuscule pour que le programme ne différencie pas majuscule et minuscule. Il faut ensuite vérifier que ce ne soit qu'un seul caractère et que ce dernier soit une lettre. Une fois que le programme s'est bien assuré que le caractère rentré par l'utilisateur est une lettre minuscule, il va voir si cette lettre est contenue dans le nom du joueur. Si elle l'est, le programme va remplacer le ou les symboles qui « cachait » la lettre dans le nom du joueur par la lettre en question. Sinon, il ne se passe rien.

Système de point

Après le processus qui vient d'être évoqué ci-dessus, le programme doit s'assurer que le nom entier n'a pas été trouvé. Cependant, s'il a été trouvé, le programme rajoute un point à l'utilisateur et recommence tout le processus pour prendre au hasard un nouveau joueur et adapter l'interface à ce dernier. Il doit à présent aussi vérifier que le nombre de points de l'utilisateur n'est pas de cinq car si c'est le cas, l'utilisateur a donc fini de trouver tous les joueurs et le programme passe soit au deuxième utilisateur soit

à la manche suivante (en fonction de la position de l'utilisateur qui a trouvé tous les joueurs).

Diverses améliorations sur la partie *pendu*

Tout d'abord, il fut nécessaire de mettre en place un timer pour éviter qu'un utilisateur fasse trop long et que cela rende l'expérience du jeu plus ennuyante. Un timer de trente secondes par utilisateur a alors été mis en place. À la fin de celui-ci, le tour de l'utilisateur se termine.

Cependant, un problème découla de ce timer. L'utilisateur n'est pas forcément prêt à jouer et le timer se déclenche immédiatement. Pour s'assurer que l'utilisateur soit prêt à jouer, des « modals » (petites fenêtres) sur lesquelles se trouve un bouton « continuer » ont été mis en place. Lorsque l'utilisateur est prêt, il appuie sur le bouton, le « modal » disparait alors et déclenche le timer.

Un autre problème qui fut primordial à résoudre fut le fait qu'un utilisateur, n'étant pas limité dans son nombre d'essais, pouvait alors « balayer son clavier » en appuyant sur toutes les touches, ce qui lui permet de trouver les joueurs en un rien de temps sans avoir à user de la moindre réflexion. Pour remédier à cela, un « système d'erreur » fut créé. L'utilisateur n'ayant seulement droit à cinq erreurs par joueur ne peut donc plus « balayer son clavier ». Comme montré sur la figure ci-dessous, les erreurs de l'utilisateur sont représentées sur l'interface par des ballons de football.

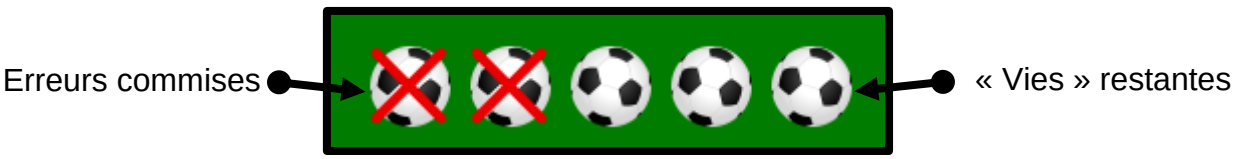


Figure 3 : Erreurs

Le dernier problème à régler était que malgré la mise en place de « modals » il était toujours possible de jouer pendant que ces derniers étaient affichés. Il fallut donc enlever le « focus » de l'input lors de l'apparition du « modal » pour remédier à ce problème.

Transmission du score

Pour transférer le score des joueurs d'un fichier à l'autre, j'ai utilisé le local storage (stockage local). C'est-à-dire que lorsque les utilisateurs ont fini leur partie, les variables sont stockées sur le serveur jusqu'à que le fichier suivant les récupère et les supprime du serveur. Cela m'a permis de transmettre des variables d'un fichier à un autre à chaque fois que j'en avais besoin.

Le terrain

Le terrain fut simplement créé avec un Canvas contenant des segments, des cercles (ou arc de cercle) et des rectangles. On peut constater que le Canvas est plus grand que le terrain lui-même, cela a deux utilités.

La première utilité vient du problème de tir. Je m'explique, si le Canvas faisait la taille du terrain et qu'un pion venait à se trouver contre un mur, il serait alors impossible de tirer à l'opposé du mur car la souris n'est pas détectée par le programme en dehors du Canvas.

La deuxième utilité est celle de l'interface, le Canvas dépassant du terrain, cela permet de mettre le score et le timer directement entre le terrain et le bord du Canvas de manière assez simple.

L'objet Cercle

Pour créer les pions et la balle, il a fallu élaborer un objet. Cet objet contient les coordonnées, le rayon, l'équipe et la vitesse du cercle. C'est grâce à l'utilisation d'objet qu'il est possible de créer et de modifier sans grandes difficultés les caractéristiques des cercles.

Composition des équipes

Après avoir reçu le score des utilisateurs, le programme crée un nombre adapté de pions et leur formation. Si on inspecte le code, on peut remarquer que quand le score d'un utilisateur se trouve entre trois et cinq, le programme créera un cercle de plus que le score de l'utilisateur. Cela est dû à un bug qui survient au premier tir. Lors de ce dernier, si l'utilisateur à plus de deux pions, un de ceux-ci sera subitement supprimé. Pour être plus précis, on peut voir dans la console qu'il existe en réalité toujours mais que ces coordonnées sont passées à « undefined » (indéfinies). Après avoir tout essayé en vain pour résoudre ce problème, il me parut judicieux de le contourner plutôt que d'y trouver une solution. Pour ce faire, j'avais remarqué que c'était toujours le même pion qui disparaissait (le deuxième s'il y a quatre pions ou le troisième s'il y a cinq pions ou plus). Pour éviter ce problème, il a donc suffi de créer d'autres cercles déjà invisibles destinés à disparaître. Ainsi, c'est grâce aux cercles en question qu'aucun pion ne disparaît lors du premier tir.

Sélection et déplacement

Pour que le programme détecte qu'on le sélectionne, il prend les coordonnées du clic de l'utilisateur et vérifie qu'elles ne sont pas dans la surface couverte par un pion. Si c'est le cas, tant que l'utilisateur n'a pas relâché le « clic », un cercle avec un trait indiquant la direction et la puissance que va prendre la balle est affiché. Ce dernier est

créé avec l'objet « Ca_cercles ». Dès que l'utilisateur relâche le « clic », le programme prend les coordonnées de sa souris et pour déterminer la distance avec le pion. Le programme va alors déterminer la vitesse du pion en fonction de cette distance. La vitesse est bien entendu limitée à partir d'une certaine distance afin que l'utilisateur n'ait ni le besoin ni la possibilité de parcourir tout le Canvas avec sa souris pour augmenter la vitesse de son pion. Suite à cela va se déclencher l'animation qui va tout simplement bouger le pion en fonction de sa vitesse et redessiner l'entièreté du Canvas toutes les huit millisecondes. Pour donner une impression de réalisme, il fut essentiel d'ajouter des frottements à cette animation. Pour ces derniers, il fut nécessaire faire un compromis entre les tirs longs ou court. Vous pouvez donc constater en jouant au jeu que les frottements sont réalistes sur des longs tirs mais qu'ils n'agissent pas vraiment sur le pion lors des tirs courts.

Collisions

Une des plus grosses difficultés dans la création de ce jeu fut la mise en place des collisions. Dans ce jeu, il y a deux types de collisions : les collisions cercle-mur et les collisions cercle-pion.

Concernant les collisions cercle-mur, le problème n'est pas si complexe à résoudre. Il suffit qu'à chaque fois que le cercle bouge durant l'animation, le programme vérifie qu'il ne soit pas en contact avec le mur. S'il l'est alors sa vitesse en x ou y est inversée (en fonction du mur). Par exemple, si le cercle touche le mur du haut, sa vitesse en y est inversée.

Pour les collisions cercle-cercle c'est beaucoup plus compliqué. Pour commencer, à chaque fois que le cercle bouge au cours de l'animation, le programme prend la position de tous les autres cercles pour vérifier les collisions. S'il y a une collision, le code va calculer la normale de collision (en d'autres termes, on peut dire qu'il calcule la « direction de la collision »). Grâce à cela, il peut maintenant calculer la vitesse relative des cercles (soit, la différence de vitesse entre les deux cercles). Ces deux données permettent ensuite de calculer la vitesse et la direction des cercles après leur collision.

Pour illustrer cela, vous pouvez observer ci-dessous une simulation de collisions faite sur *GéoGebra*. (Cette simulation est à votre disposition dans le fichier collisions.ggb) :

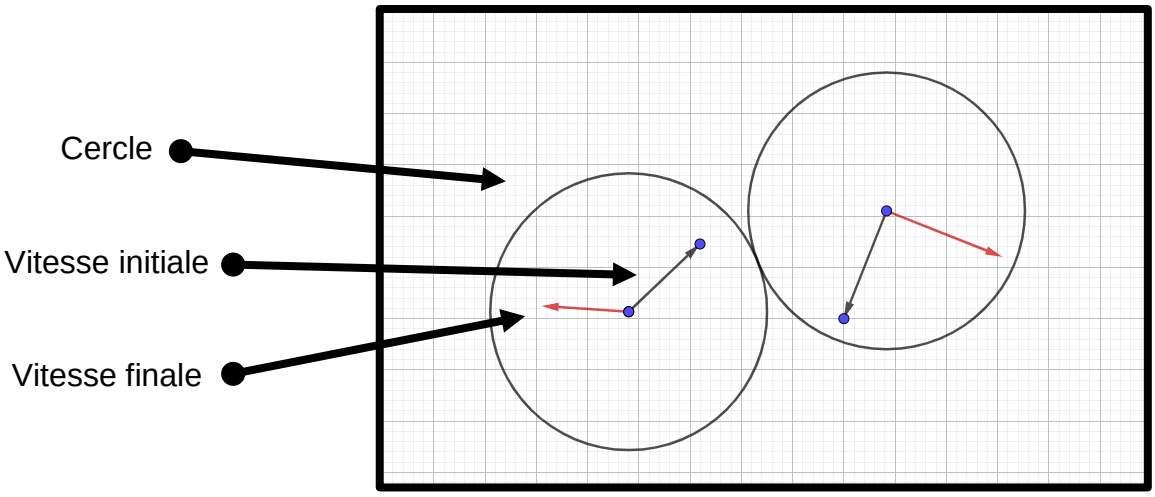


Figure 4 : collisions

Diverses améliorations sur la partie *Table soccer*

Comme à la manche précédente. L'ajout d'un timer parut nécessaire au dynamisme du jeu. Il y a donc maintenant un timer de 13 secondes qui a été ajouté et il est possible de le voir à côté du score de l'utilisateur qui doit jouer.

Figure 5 : timer

Un autre ajout servit à la résolution d'un bug. Lorsqu'un utilisateur tirait, l'autre n'avait pas besoin d'attendre que l'animation de déplacement se termine pour tirer à son tour. Ceci crée donc un bug où deux timer (ou plus) fonctionnent en même temps et « se marchent sur les pieds » entrainant des réactions totalement inattendues (pion qui n'avance plus en général). Pour corriger cela, il a été fait en sorte que lorsqu'un utilisateur souhaite tirer, le code ne vérifie qu'aucun autre cercle ne soit en mouvement. Si c'est le cas, l'utilisateur ne peut pas tirer.

Dans le même style de problème, il y a celui des équipes. Pour éviter qu'un utilisateur puisse jouer les deux équipes, il a été décidé de remplacer le contour noir des cercles par un contour jaune sur les pions de l'utilisateur qui doit jouer. Les deux fonctions de ce changement sont tout simplement de simplifier la compréhension du jeu et d'indiquer au code quels sont les pions jouables.

Bugs sur la partie *Table Soccer*

Malgré toutes les modifications effectuées pour réduire les bugs de collisions, il arrive encore qu'ils surviennent même si cela est devenu relativement rare. Les cercles peuvent donc se coincer dans un mur ou dans un autre cercle de temps en temps.

Selon moi, le problème vient du nombre de « détection de collision » effectuées. Le cercle peut bouger de jusqu'à dix pixels par image, cela implique qu'il peut sur une image n'être en contact avec rien et l'image d'après se retrouver coincé dans un autre cercle.

FONCTIONNEMENT DU JEU EN LIGNE

Communication joueur-serveur

Lors de la création d'un jeu online, il y a plusieurs manières d'établir la communication entre les joueurs et le serveur. La plus courante est la mise en place de « web sockets ». Comme illustré sur l'image ci-dessous, ces derniers permettent une communication bidirectionnelle avec le serveur. Ceci augmente donc drastiquement la vitesse des échanges, la sécurité et la stabilité du serveur. Malheureusement, je n'ai pas réussi à faire l'usage de web socket et il a fallu se rabattre sur une autre méthode, l'utilisation d'AJAX pour établir un canal de communication « half-duplex » (semi-duplex). Ceci consiste à utiliser AJAX et PHP pour faire des requêtes à une certaine récurrence. Même si l'utilisation de cette méthode réduit grandement la vitesse des échanges avec le serveur, étant donné que le jeu n'est pas en temps réel mais en tour par tour, cela ne pose pas trop de problèmes. Pour simplifier la compréhension de ces deux méthodes de communication, voici ci-dessous deux schémas tirés d'une conférence de Jeff Kolesnikowicz.

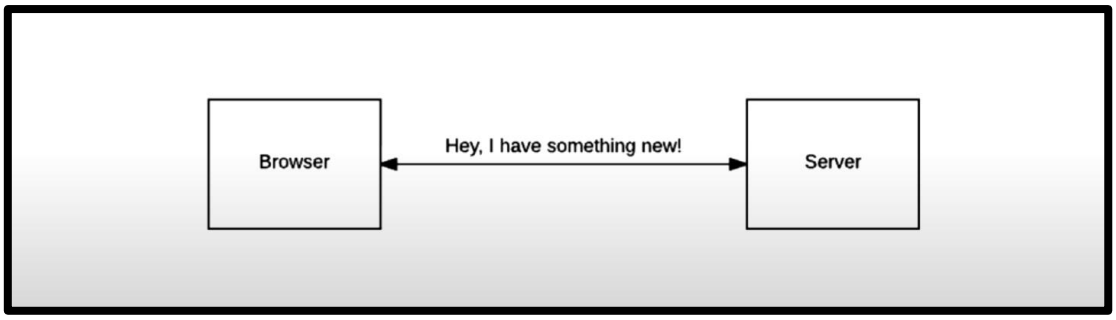


Figure 6 : Websockets

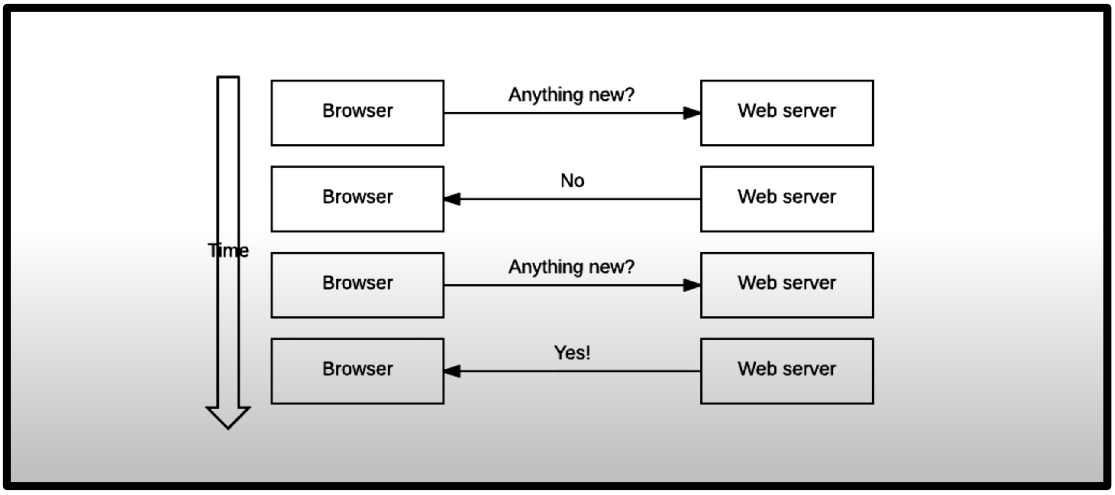


Figure 7 : AJAX half-duplex

Création de rooms

Une fois que l'utilisateur a appuyé sur le bouton « En ligne », il arrive sur une page qui lui demande de rentrer son nom d'utilisateur et de « Rechercher ou créer une room ». Une fois ces deux choses faites, le code récupère le nom d'utilisateur choisi et l'envoie vers un fichier PHP avec une requête AJAX. Le fichier PHP regarde les rooms disponibles. S'il n'en trouve aucune, il en crée une. Sinon, il vérifie le nombre d'utilisateurs dans celle-ci. Si aucune room n'a moins de deux utilisateurs, il en crée une. Sinon, l'utilisateur rejoint la room n'ayant qu'un seul utilisateur. Une fois que le deuxième utilisateur est dans la room les deux utilisateurs sont envoyés vers la partie *pendu*.

Mais un problème survient alors. Le premier utilisateur ayant rejoint la room n'est pas emmené vers la suite du jeu. La raison est simple, le programme n'utilise pas de websocket. Comme expliqué plus tôt, le seul moyen (à ma connaissance) de faire en sorte que le serveur envoie un message au joueur est d'utiliser un websocket, ce qui n'est pas le cas dans ce programme. Pour remédier à cela, le programme envoie une requête au serveur toutes les secondes pour être mis au courant de l'état de la room. Lorsque celle-ci est prête, l'utilisateur est envoyé vers la manche suivante. Mais l'utilisation de cette méthode implique qu'entre le temps que le deuxième utilisateur soit envoyé vers la manche suivante et le premier, il peut y avoir jusqu'à une seconde.

Partie *pendu* de local à en ligne

Le programme de la partie *pendu* a bien entendu dû être modifié entre le mode *Deux joueurs* et le mode *En ligne*. Tout d'abord, il fallut récupérer l'ID de l'utilisateur ce qui fut utile plus tard lors de l'échange des scores avec l'adversaire. Pour faire cela, j'ai réutilisé le stockage local du serveur.

Par la suite, il fallut refaire l'entièreté de la structure du jeu afin qu'il soit jouable par un joueur seul.

L'échange des scores

Pour que les utilisateurs s'échangent leur score il faut, comme précédemment, faire une requête AJAX qui renvoie vers un fichier PHP le score et l'ID de l'utilisateur. Ce fichier rentre ensuite ces informations dans une table MySQL pour les rendre accessibles à l'autre utilisateur.

Pour que l'autre utilisateur les récupère, il faut tout d'abord attendre cinq secondes car comme dit précédemment, il y a un décalage entre les deux utilisateurs. Ceci permet de laisser le temps aux deux utilisateurs de déposer leur score sur le serveur. Une fois ces cinq secondes passées, le code commence donc par rechercher l'ID de l'adversaire. Pour cela, il suffit qu'il utilise l'ID de l'utilisateur pour retrouver sa room et

donc son adversaire. Une fois l'ID de l'adversaire trouvé, il va chercher quel est le score déposé sur le serveur par ce dernier et le renvoyer vers le fichier initial.

Transmission des scores entre les fichiers

Un problème assez logique mais plutôt inattendu fit surface ici. Précédemment lorsqu'il fallait transférer des données d'un fichier à un autre, l'utilisation du stockage local fonctionnait. Mais le problème ici est qu'un utilisateur arrive avant l'autre pour récupérer ces données. Cela implique que lorsque le premier utilisateur envoie les scores, ils sont ensuite modifiés par le deuxième utilisateur. Cela implique aussi que le premier utilisateur arrivé sur la partie *Table Soccer*, lorsqu'il supprime les variables du serveur, les rend inaccessibles à son adversaire. Concernant ce deuxième problème, il aurait suffi d'attendre un certain temps avant de supprimer les variables du serveur. Mais cette méthode ne fut pas appliquée à cause du premier problème. Pour mieux saisir en quoi cela pose problème, il faut comprendre comment fait le code pour détecter quel utilisateur possède quels pions. C'est en fait relativement simple, chaque utilisateur possède les pions de « l'équipe 1 » (bleu) selon son point de vue. Cela se fait car selon un utilisateur le score est de « X - Y » et selon l'autre il est de « Y-X ». Voilà pourquoi le fait que dans le stockage local, il n'y ait qu'un score de disponible pose problème. Car les deux se retrouvent donc avec « Y-X » et jouent les mêmes pions.

Pour remédier à cela, il fallut transmettre des données via l'URL de la page web. Quand un utilisateur arrive vers la partie *Table Soccer*, son url indique au code s'il est le joueur un ou deux (en fonction de son id). Grâce à cela, chaque joueur peut récupérer sa « version » du score, son ID et celui de son adversaire facilement car ces données sont enregistrées dans le stockage local de manière séparée pour que chacun des deux utilisateurs récupère ce dont il a besoin.

Amélioration sur la partie *pendu*

Tout d'abord, il fallut évidemment faire disparaitre l'input de l'écran une fois le temps écoulé ou les cinq joueurs trouvés. Ceci sert évidemment à empêcher l'utilisateur d'augmenter son score alors que la manche est censée être finie. Cela sert aussi à éviter un décalage entre les deux utilisateurs. Pour faire simple, si l'input ne disparaissait pas, l'utilisateur enverrait son score au serveur à la fin du temps réglementaire mais pourrait continuer à le modifier durant les cinq prochaines secondes. L'autre utilisateur ne recevrait alors pas ce score-ci et un décalage se créerait entre les deux utilisateurs.

Partie *Table Soccer* de local à en ligne

Comme pour la partie *pendu*, il faut faire en sorte que le jeu soit jouable par un utilisateur seul. Comme expliqué précédemment, il faut donc corriger le programme pour que l'utilisateur ne puisse jouer que les pions bleus et que l'adversaire soit représenté par les pions rouges.

Transmission des tirs

Pour transmettre les tirs entre les utilisateurs, c'est basiquement comme les autres échanges avec le serveur. Un utilisateur envoie des données au serveur. L'autre utilisateur regarde à une certaine récurrence s'il y a les données qu'il recherche sur le serveur. Une fois qu'il les a reçues, il les implante dans son code et ainsi de suite jusqu'à la fin de la partie. En l'occurrence, l'utilisateur envoie au serveur le numéro du pion qui a été tiré, et la position de la souris par rapport à celui-ci. Cela permet au code de l'autre utilisateur d'avoir toutes les données nécessaires à un tir.

Bugs sur la partie *Table Soccer*

Ayant la même manière de gérer les collisions qu'en local, il n'échappe par conséquent pas aux bugs de collisions. Ces derniers sont cependant largement plus handicapants ici car les bugs ne se produisent pas forcément chez les deux utilisateurs ce qui peut donc les désynchroniser complètement. Il arrive aussi que les utilisateurs soient désynchronisés sans raison particulière. Malheureusement, je n'ai ni trouvé la cause de ce problème ni la manière de le résoudre.

Amélioration sur la partie *Table Soccer*

J'ai quand même réussi à limiter ces désynchronisations en limitant le nombre de décimales des données utilisées à quatre. De cette façon, on résout une manière que le jeu avait de se désynchroniser. Cette dernière était un décalage qui se créait et augmentait au fil de la partie jusqu'à être totalement désynchronisé. La cause de ce décalage croissant était que le nombre de décimales des données que le serveur stockait et envoyait à un utilisateur était de quatre tandis que celui de l'autre était de huit (car les données ne passent pas par le serveur pour lui). Cela a drastiquement réduit le nombre de bugs de désynchronisation même si celui-ci reste élevé.

DEVELOPPEMENT FUTUR

Interface

Concernant l'interface, il y a encore trois points à développer :

Premièrement, il faudrait tout simplement la rendre plus belle. Malgré avoir bien travaillé sur ce point, le ressenti d'une interface faite par un débutant est encore trop présente à mon goût.

Deuxièmement, quel que soit le travail fourni sur l'interface, si elle n'est pas « responsive », c'est-à-dire qu'elle s'adapte au format de l'écran de l'utilisateur, elle sera quand même disproportionnée sur certains écrans. Ce point-là est extrêmement important car dans le monde actuel, la plupart des visiteurs de site web sont sur smartphone, chose à laquelle l'interface actuelle n'est pas du tout préparée.

Troisièmement, la musique contribue grandement à créer l'ambiance d'un jeu. L'absence de celle-ci est donc un manque à combler au plus vite.

Modes de jeu

Pour rendre le jeu moins répétitif et lassant, l'ajout de plusieurs modes de jeu est indispensable.

La première idée serait l'ajout d'un mode « deux contre deux ». Un mode ou l'on joue par équipes de deux pourrait rendre le jeu grandement plus attrayant et ne semble à première vue pas si compliqué que cela à mettre en place.

Le deuxième mode à ajouter serait le « tous contre tous ». Un mode à quatre joueurs, chacun pour soi. Même si c'est sûrement beaucoup plus compliqué à mettre en place que le « deux contre deux » cela pourrait encore plus diversifier le jeu. Ce mode ressemblerait à l'image ci-dessous.

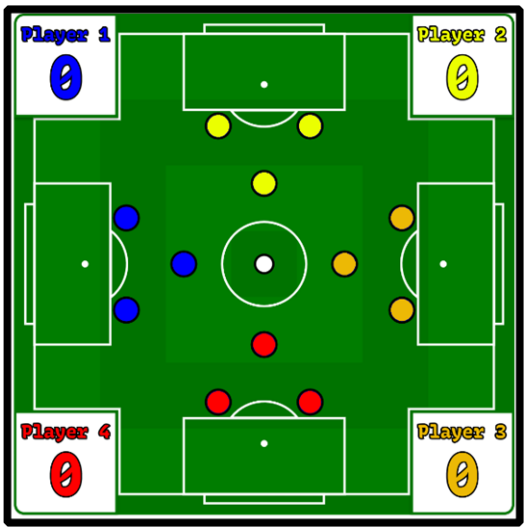


Figure 8 : Concept

Le dernier mode de jeu, serait le mode « chaos ». Ce serait comme le mode basique (un contre un) sauf qu'au lieu du tour par tour, chaque joueur peut tirer quand il veut. Il n'aurait donc pas à attendre le tir de l'adversaire et cela donnerait lieu à un mode assez chaotique mais qui peut renouveler complètement l'expérience du jeu. Cependant, il serait compliqué de le mettre en place car un mode en temps réel comme cela nécessite l'utilisation de websockets.

Divers

Rien ne pousse plus l'Homme à se dépasser que la compétition avec ses semblables. Ce n'est pas pour rien que la plupart des jeux vidéo actuels possède un système de classement, pourquoi en serait-il différent pour celui-ci. L'ajout d'un système de classement entre les utilisateurs me semble donc être une excellente idée qui donnerait aux joueurs un objectif à atteindre.

CONCLUSION

Je souhaiterais donc conclure ceci avec les choses que j'ai tirées de ce travail de maturité et qui resteront avec moi pour le restant de mes jours.

Pour commencer, ce travail m'a bien évidemment énormément appris sur le plan de la programmation. Mais comme vous le savez, la programmation ne se résume pas à notre connaissance d'un langage ou d'un autre. C'est une question de persévérance, d'analyse de nos erreurs, de recherche, de remise en question et de tant d'autres choses pour enfin voir apparaître sous nos yeux, le produit de notre imagination et de notre dur labeur.

Mais sur un travail de cette envergure, il ne suffit pas de se passionner à la programmation le temps d'une semaine acharnée. Il faut anticiper la charge de travail que notre projet impliquera, la répartir intelligemment sur toute la durée mise à disposition et donner le meilleur de soi-même pour se tenir à ce programme établi.

Si malgré tout on réussit à faire tout cela, on en ressort alors grandi et rempli de fierté. Cette fierté c'est celle d'avoir su imaginer un projet qui nous faisait rêver (tout en restant réaliste) et de l'avoir concrétisé malgré les embuches.

C'est pour cela que je tiens à clore ce rapport en remerciant mon mentor, qui m'a guidé dans ce travail. Je tiens aussi à remercier « le Lycée » de m'avoir donné l'opportunité de réaliser ce projet au travers du travail de maturité. La création d'un jeu vidéo était un rêve d'enfant, qui comme tant d'autres, n'aurait jamais été réalisé si je n'avais pas été contraint de le faire par ce travail.

SOURCES

Animations

- Csslab, https://csslab.app/animations/wobble , Août 2023
- Csshint, https://csshint.com/confetti-falling-background-using-canvas, Août 2023

WebSockets

- KOLESNIKOWIC Jeff, Realtime PHP Using Websockets, https://www.youtube.com/watch?v=Q7Us_DjMbXU&ab_channel=CakePHP, Octobre 2023
- CyberWolves, https://www.youtube.com/watch?v=1fjICYqfUG4&ab_channel=CyberWolves, Octobre 2023

Collisions

- ChatGPT, https://chat.openai.com , Mai 2023 et Octobre 2023

Interface

- Gravity soccer, https://poki.com/fr/g/gravity-soccer, Avril 2023
- Penalty superstar, https://poki.com/fr/g/penalty-superstar, Avril 2023
- Foot-chinko, https://poki.com/fr/g/foot-chinko, Avril 2023
- Basti Ui, https://www.youtube.com/watch?v=I53D9EeJiqs&ab_channel=BastiUi, Octobre 2023

Programmation

- W3schools, https://www.w3schools.com/js/js_ajax_intro.asp, Octobre 2023
- W3schools, https://www.w3schools.com/php/default.asp, Octobre 2023
- BRUCHEZ Christophe, Chapitre 6 - objets, Avril 2023