TELECOM SudParis

Théo Danielou
Rohitkumar Datchanamourty
Brieuc Popper
David Zhu

Projet informatique 1ère année PRO3600

DRAW2.IO

Enseignante responsable: Chantal Taconet

SOMMAIRE

1 Introduction **p.3**

2 Cahier des charges **p.4**

3 Développement **p.7**
3.1 Analyse du problème et spécification fonctionnelle p.7
3.2 Conception préliminaire p.8
3.3 Conception détaillée p.9
3.4 Manuel utilisateur p.12

4 Conclusion **p.13**

A Annexe **p.14**

Chapitre 1

Introduction

Ce projet est mené par une équipe de 4 étudiants en première année de Télécom SudParis dans le cadre du module PRO3600 avec comme enseignante responsable du suivi de ce projet, Chantal Taconet.

Ce document présente dans un premier temps le cahier des charges, puis le déroulé du développement des différents modules ainsi qu'un guide d'utilisation. On conclura en abordant les différentes pistes d'évolutions possibles.

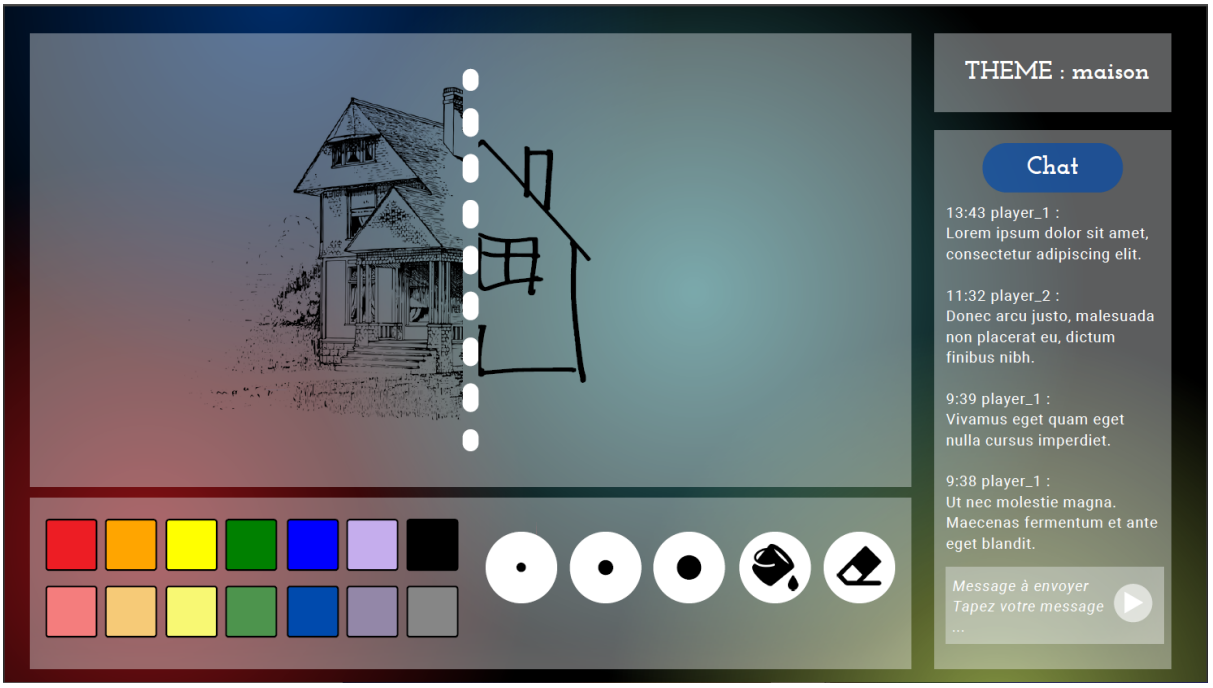
Le projet est de réaliser un jeu multijoueur sur navigateur web. Il s'agit d'un jeu de dessin compétitif qui fait s'affronter les joueurs sur plusieurs manches. A chaque manche, les joueurs doivent créer en un temps imparti, et suivant un thème imposé, le dessin le plus qualitatif possible. La subtilité est que chacun ne peut dessiner que sur une moitié de l'écran et doit collaborer avec un autre joueur de la partie, afin de créer un dessin en duo. Les duos changent à chaque manche. Il s'ensuit alors une phase finale de vote où les joueurs attribuent des notes aux dessins, avant que ne soit révélé le classement final. Le but est, en plus de s'amuser bien-sûr, d'être le mieux classé possible. Cela induit donc une bonne coopération des duos à chaque manche.

Chapitre 2

Cahier des charges

Il s'agit de développer un jeu multijoueur en ligne sur navigateur web. Il se destine donc à des utilisateurs dotés d'une connexion internet et d'un ordinateur.
L'objectif principal est d'achever une version du jeu jouable avec une implémentation complète du processus de jeu.
Plusieurs jeux similaires existent déjà. On pourra citer Gartic Phone ou Skribbl.io qui nous ont inspirés.

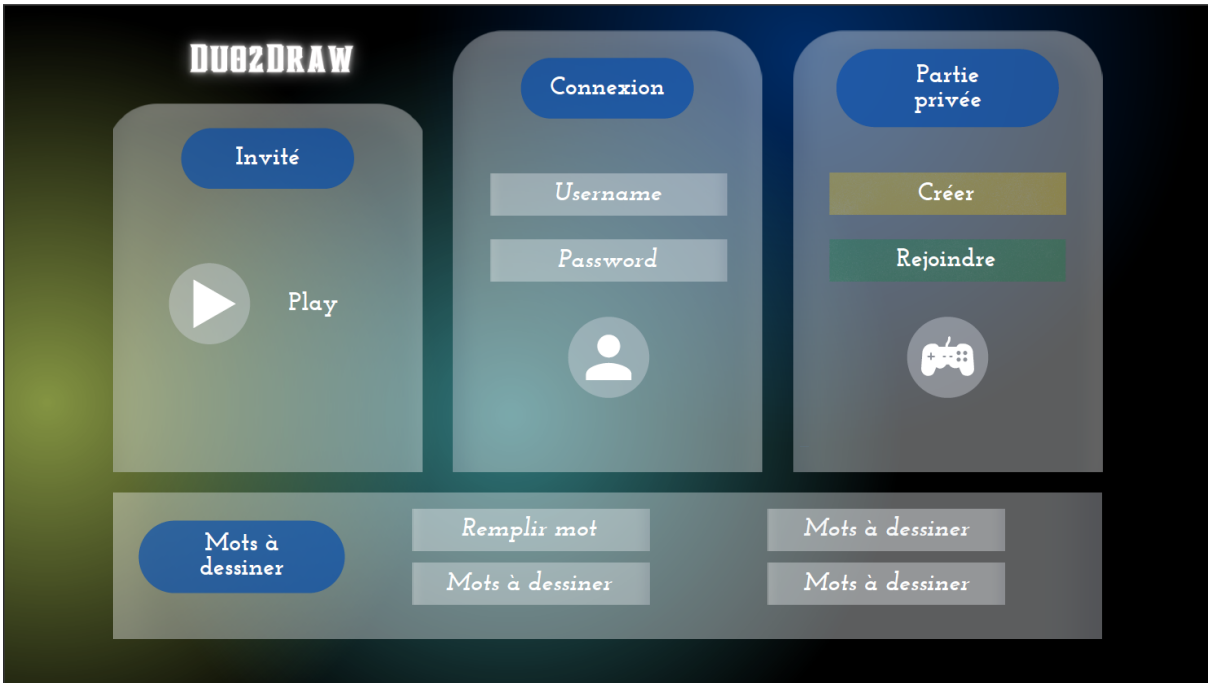
LISTE DES FONCTIONNALITÉS ATTENDUES



L'interface finale de l'application devra s'apparenter à cette maquette.

- Interface graphique "en jeu" : choix de la couleur, du pinceau, possibilité de dessiner sur une partie de l'écran allouée.
- Interface "phase des votes" : les joueurs voient tous les dessins faits lors de la partie et attribuent un score à chacun.
- Interface "résultats" : tous les joueurs prennent connaissance du vainqueur et des scores de chacun.
- Création de parties privées, accessibles uniquement via un lien par exemple.

- Interface d’accueil



Maquette de l’écran d’accueil

LISTE DE FONCTIONNALITÉS BONUS

- Systèmes de “chat” soit vocaux soit textuels en temps réel
- Authentification des joueurs (création d’un compte avec mot de passe, historique des parties, scores…)
- Création de files d’attentes pour un système dit de “matchmaking” (création de parties entre plusieurs groupes d’individus trop peu nombreux pour faire une partie à eux seuls)
- Ajout de tournois

LES CHOIX TECHNIQUES

- Côté “Front-end” (en local sur le navigateur d’un joueur)
    - Site web dynamique codé avec HTML, CSS et Javascript
    - Librairie p5.js (voir annexe) afin d'accélérer la création des outils de dessin

- Côté “Back-end” (le serveur):
    - Serveur codé en Python avec la libraire Flask (voir annexe) qui est un framework dédié à la création d’applications web.

    - Communication entre le front-end et le back-end grâce à la librairie Socket-IO (voir annexe) qui permet la communication en temps réel entre les deux parties sur la base d'événements déclenchés.
    - En cas de conception d’un système d’authentification utilisateurs, une base de donnée pourra être implémentée à l’aide de SQLite (voir annexe)

- Déploiement sur Internet : avec Heroku ou mineT. Les deux services permettent d’héberger des applications web.

Chapitre 3

Développement

3.1 Analyse du problème et spécification fonctionnelle

Quel est le déroulement exact de la phase “dessin” ?
Chaque joueur peut choisir différentes couleurs à l’aide d’une palette, différents pinceaux et dessiner dans une zone qui lui est réservée avec une souris ou un autre dispositif d’entrée. Le temps est limité, un timer informe les joueurs du temps restant. Une zone en haut affiche le nom du binôme qui dessine avec le joueur et le thème actuel sur lequel réaliser le dessin.
Quel est le déroulement exact de la phase “vote” ?
Chaque dessin apparaît successivement pendant un temps limité sur l’écran de tous les joueurs, sans que les auteurs du dessin soient cités. Le thème est rappelé, et les joueurs peuvent laisser une note entre 0 et 10 au dessin en question. Le serveur enregistre les notes qu’il utilisera ensuite pour calculer les scores.

Quelles sont les informations disponibles par profil joueur?
Le profil joueur contient un nom, des infos personnelles (description, photo de profil) et des informations liées à l’historique des parties déjà jouées (scores, consultation d’anciens dessins…)

Quelles sont les possibilités en nombre de joueurs ?
La première fonctionnalité que nous allons créer sera de créer une première partie privée avec 4 joueurs différents. Chacun de ces 4 joueurs devra via un code secret se connecter à ce “salon privé”
Plus tard nous mettrons en place un système qui permet à n joueurs (avec $n < 4$) de se mettre en recherche de partie et de compléter ce nombre de joueurs par d’autres joueurs qui cherchent une partie en même temps afin de créer des parties.

Quelles fonctionnalités essentielles doivent être présentes dès les premiers prototypes?

Le projet étant assez ambitieux étant donné les heures fixées pour ce projet et notre expérience de développement web, nous avons une hiérarchie de fonctionnalités plus ou moins “bonus”, que nous ajouterons éventuellement si nous avons le temps et si les

fonctionnalités plus essentielles fonctionnent correctement. Ainsi la création d'une partie privée avec tous les joueurs qui se connectent à un même salon, puis une phase de dessin avec plusieurs mots et plusieurs binômes de dessins, puis une phase de vote basique constituent l'objectif à atteindre dès que possible. Puis nous ajouterons éventuellement des fonctionnalités et des détails plus intéressants et avancés.

3.2 Conception préliminaire

L'application se découpera en 6 modules avec une interdépendance plus ou moins forte:

- ***front-end***: désigne la conception d'une maquette graphique du site et la programmation de l'ensemble des interfaces graphiques permettant à l'utilisateur d'interagir sur le site tout au long de son expérience

- ***système d'authentification des comptes utilisateurs (login)***: dédié à l'identification des utilisateurs lors de leur connexion

- ***matchmaking***: comprend le système de gestion de la queue des joueurs en recherche de partie ainsi que la création d'une nouvelle salle de jeu lorsque suffisamment de joueurs sont trouvés

- ***étape de dessin (pendant une partie)***: désigne la première partie du jeu où les joueurs dans la salle dessinent simultanément sur une zone commune

- ***étape de vote (pendant une partie)***: correspond à la deuxième partie du jeu où les joueurs votent pour le meilleur dessin parmi tous ceux dessinés

- ***chat utilisateur***: dédié au système de communication par écrit pour les joueurs dans une même salle

Tandis que le module front-end peut être produit indépendamment des autres (il n'y a pas de à vraiment savoir comment est implémenté le back-end pour coder l'interface, c'est plutôt le back-end qui s'adapte à l'organisation du front-end), le développement des autres modules devra se faire de manière relativement séquentielle. Le découpage produit est en effet plus simple à gérer vu notre manque d'expérience, d'autant plus qu'il suit l'expérience de l'utilisateur dans l'ordre chronologique.

Le système d'authentification implique la création d'une base de données en back-end pour stocker les informations des comptes utilisateurs.On pourra opter pour une structure de données stockée sur le serveur avec laquelle on communiquera en SQL.

Le matchmaking devra gérer la mise en attente des joueurs lors de la recherche d'une partie et l'affectation d'un groupe de joueurs dans la file d'attente à une nouvelle salle. On privilégiera ainsi une file FIFO.

L’étape de dessin nécessite la représentation en temps réel et en simultané des éléments dessinés par les joueurs sur chacun des écrans. Ce module est assez dépendant du front-end.

De manière similaire, l’étape de vote requiert la simultanéité de l’affichage et la centralisation des résultats de vote.

Le chat utilisateur doit attendre une ébauche au moins de l’interface graphique pour être développé.

Planning prévisionnel

Février	**Mars**	**Avril**	**Mai**
front-end (pour l'authentification et le matchmaking) système de login Livrable 1 Formations (interface graphique, authentification, base de données)	front-end (pour l’étape de dessin, de vote et le chat) gestion de l’étape dessin Formations (communication du dessin en temps réel)	gestion matchmaking étape de vote Formations (matchmaking)	chat utilisateur déploiement serveur Formations (chat utilisateur, déploiement)

3.3 Conception détaillée

La première étape est de mettre en place un serveur Flask afin d’afficher les pages web développées.
Il a fallu réfléchir au problème du matchmaking: comment créer des parties à partir des joueurs en recherche de partie. On a donc déplacé le problème vers un système de salles: il y a un certain nombre de salles de jeu ouvertes et les joueurs souhaitant jouer peuvent rejoindre l’une de ces salles depuis la page d’accueil. Lorsqu’une salle contient suffisamment de joueurs, une partie démarre automatiquement. Le joueur a alors la possibilité de quitter ou rejoindre une salle lorsqu’il le souhaite.
En parallèle, le développement de l’outil de dessin a pu se faire assez rapidement à l’aide de librairies. Les utilisateurs étaient à ce stade capables de tracer leur dessin sur le navigateur. Cependant, celui-ci n’était pas encore diffusé aux autres joueurs sur leur propre navigateur. On a donc commencé l’implémentation des sockets qui permettent de récupérer les coordonnées des nouveaux points tracés, les faire passer par le serveur puis les renvoyer aux autres joueurs présents dans la partie. Cela a induit la conception de fonctions à l’écoute des événements utilisateurs (ex: appui sur le bouton de la souris, etc…).

Pour le déploiement du serveur, on a opté pour un hébergement sur le serveur Minet. L'association du campus minet permet, pour les cotisants, de créer des machines virtuelles personnelles qui ont une adresse IP propre.
Ces serveurs Debian 11 VM sont donc des serveurs linux accessibles par ssh et la ligne de commande. Sur ce serveur on a un site web qui est hébergé via nginx, et 10 serveurs flask pythons qui sont lancés manuellement et qui permettent chacun de gérer une "room". Ainsi le site web central statique plus les 10 serveurs (on pourrait en avoir un nombre quelconque) sont hébergés sur cette machine virtuelle.
On a pu cloner nos fichiers via github sur le serveur et ensuite démarrer les serveurs.

Pour la partie graphique, nous avons tout d'abord analysé les sites, comme gartic phone, qui existent déjà pour en tirer des idées sur la conception générale de la charte graphique.

Il faut noter que lors de la conception du site web, nous avons opté pour un design responsive qui correspond à un mode de développement de site qui s'adapte à la taille des écrans d'ordinateurs et de téléphone. Pour cela, après avoir fini une première version pour la taille d'un écran de l'ordinateur, nous avons utilisé les requêtes média (plus connus en anglais : media queries) afin de rendre le site aussi agréable à voir sur un ordinateur que sur un smartphone.

En parallèle, pour une meilleure adaptation à la taille des écrans possibles, nous avons fait usage de flex box. Les deux solutions possibles étaient les flexbox ou les grid, mais les grid qui sont plus simples à construire sont moins adaptatif par rapport à la taille des écrans. D'ailleurs c'est bien ce que l'on retrouve dans ce projet, lorsque nous avons glissé un grid dans une flex blox. D'ailleurs ci-contre se trouve la disposition des boîtes dans le site :

Aussi, un des points à travailler à été de rendre le site plus vivant et plus compréhensible par tous, d'où l'usage de hover effect pour changer les liens cliquables en des boites cliquables.
Étant donné qu'on souhaite véhiculer l'idée d'un jeu amusant, nous avons utilisé un fond dynamique pour rendre le jeu attrayant et neuf. La technique employée derrière est une

grande image avec des couleurs définies par un gradient en background et un animation réalisé à l'aide d'une règle keyframes qui a défini un mouvement en boucle et sur une durée de dix secondes.

La conception a été réalisée sans l'utilisation de frameworks en HTML ou CSS comme Bootstrap. En effet, le développement web était un accès à de nouveaux langages de programmations, et apprendre à utiliser les propriétés et fonctions natives s'ils paraissent suffisant pour développer à bien le projet avec le temps fourni.

Les plans de charges prévisionnel et final sont disponibles en annexe (se référer au fichier PlanDeCharges.pdf positionnés dans le même dossier que ce rapport).

3.4 Manuel utilisateur

Pour lancer le serveur en local : lancer le script app.py avec un argument i, pour etre sur le port 5000+i. ("python3 app.py 1" pour le port 5001 par exemple)
Puis lancer roomi.html (dans Projet final, puis client) (par exemple room1.html)

Une fois sur la page web, le joueur doit entrer dans la file de recherche de partie, lorsque 4 joueurs sont dans cette file la partie peut débuter. Il faut aussi choisir un nom lors de la recherche de partie, ce nom doit être différent des autres noms des gens présents.

Une fois dans la partie, chaque joueur peut avec le clic gauche de sa souris dessiner sur sa moitié d'écran et tenter de faire le meilleur dessin possible, sachant qu'il ne peut pas sortir de sa moitié et doit donc coopérer avec son partenaire du moment.

Les équipes de deux changent au cours de la partie, puis une fois que toutes les équipes possibles ont fait un dessin, la phase de vote débute.

Lors de cette phase, un joueur peut cliquer sur les boutons pour donner son avis sur les dessins, et chaque joueur participant à un dessin gagne des points en fonction des votes positifs que son dessin reçoit.

Pour jouer en ligne : aller sur l'url draw2io.h.minet.net, se référer au manuel ci-dessus.

Chapitre 4

Conclusion

La réalisation concrète de ce projet a induit la modification de certains de choix initiaux en termes de technologies utilisées mais également en termes d'architecture (ex: le matchmaking) pour contourner des difficultés techniques.
Plusieurs problèmes sont également apparus: le tracé des lignes n'est pas continu et s'explique vraisemblablement par une limitation de la librairie utilisée pour les sockets.
De nombreuses perspectives d'évolution sont ainsi envisageables: on a par exemple été contraints à une duplication des fichiers HTML pour les différentes salles de jeu, alors qu'une factorisation serait possible avec un peu plus de travail. Une gestion dynamique du nombre de salles serait également envisageable pour s'adapter en temps réel au nombre d'utilisateurs sur le site. D'un point de vue esthétique, les interfaces sont améliorables afin d'offrir quelque chose de plus qualitatif. Une mise en place de système d'authentification des utilisateurs pourrait notamment permettre de garder un historique des scores des meilleurs joueurs. Nous avons par ailleurs renoncé au chat utilisateur lors du développement mais cela constituerait une idée bienvenue dans un jeu en ligne.

Annexe

[1] p5.js	https ://p5js.org/
[2] socket.io	https ://socket.io/
[3] flask	https ://flask.palletsprojects.com/en/2.1.x/
[4] SQLite	https ://www.sqlite.org/