**Equipe :**

* Hien Minh NGUYEN : expert graphisme
* Morgan FEURTE : expert web
* Nathan BONNARD : expert Unity
* Oummar MAYAKI : expert Puzzle & Dragons
* Alban DESCOTTES : expert Triathlon

**Nom de l’équipe :** Objectif™ Lune™

**Idée de projet :**

*Synopsis :*

Un site web disposant d’un système de comptes qui contient un jeu intégré dans le navigateur. Le jeu donne un score à chaque joueur (ou permet les 1vs1) ce qui donne lieu à une statistique dans la base de données liée à chaque compte, permettant ainsi de faire un classement des meilleurs joueurs.

*Définition du jeu :*

To be reflexionnisated

Domaines à inclure dans le jeu :

- IHM

- Webdesign

- Réseaux et multijoueur

- Moteur physique

**Différentes idées de jeu :**

* Nathan :

**Type de jeu :** Survival - Grille - Dofus ?

**Contrôles :** Souris

**Synopsis :** Tu crées un joueur que tu personnalises avec des items et des caractéristiques pour ensuite le lancer dans une arène où il devra être le dernier survivant (jeu en ligne multijoueur). L’arène sera sous forme de secteurs côtes à côtes qui auront des caractéristiques random (coffres contenant des items, plans d’eau pour récup…) implémentés au début de chaques parties (génération de la carte aléatoire).

Le joueur devra se placer sur une case au début de chaque survival sans avoir connaissance de la position des autres joueurs, cette prise de décision sera donc stratégique (plutôt aller vers un coffre mais risque de mourir entraîné dans un combat ou se placer loin de tout pour être le dernier en vie et viser le bonus de fin de game). Il pourra ensuite se déplacer à chaque “tour” et s’il rencontre un coffre, il récolte les items, s’il rencontre un autre joueur → début du combat qui se passe automatiquement en fonction des items et caractéristiques.

**Intérêts de votre jeu à être en ligne (score ? multijoueur ?) :** le jeu devra forcément être multijoueur mais peut-être implémentation d’IA pour jouer tout seul si personne n’est connecté ?

* Minh :

**Type de jeu :** 2D Vue de haut

**Contrôles :** Flèches directionnelles/ZQSD

**Synopsis :** s u r r e a l vaporwave imaginary visual-heavy tripping binaural experience

**Intérêts de votre jeu à être en ligne (score ? multijoueur ?) :** intérêt expérimental

* Alban :

**Type de jeu :** Plateforme 2D

**Contrôles :** ZQSD déplacement, OKLM shoot/jump/switch weapon/action

**Synopsis :** Plusieurs niveaux dans plusieurs environnement différent : Sur terre, sur la lune (gravité changé). Le perso avance saute etc ... kill w/ gun shotgun etc avec carac précises, ces armes influent sur ta vitesse, la hauteur de tes sauts. Tu peux sauteur plus haut pour pas aller sur des plateformes et pas être vu par exemple, ou récupérer de la vie. Comme ca tu peux pas faire tout le jeu avec les mêmes armes.

Le niveau de difficulté influe sur ta vie, le nombre de soins trouvables les dégâts des armes ennemies, le nombre d’ennemies peut être...

Côté scénario : peut être un mec qui est genre espion à la base, il espionne (CQFD), tombe sur des expériences douteuse, et bam des zombies. Donc il change de mission et cherche à résoudre le truc. Il va sur des lieux différents pour sauver le monde ! et là on peut dire que la conspiration à la base c’est la “s u r r e a l” (pour avoir le vote de Minh).

**Intérêts de votre jeu à être en ligne (score ? multijoueur ?) :** Classement sur chaque niveau du chrono, chrono total. Niveau difficulté etc

Et niveau bonus quand tu as fini, avec masse de zombies qui arrive et tu dois en tirer le plus possible. avec bien sur une carte aléatoire (vote de Nathan).

Je sais pas si c’est trop ambitieux ou non, mais ca peut donner un aspect intéressant sur l’écriture car vous êtes assez inventif, on peut même caler une action finale sur un trébuchet (vote de Oummar et Morgan).

* Oummar :

**Type de jeu :** Shoot’em up en scrolling vertical

**Contrôles :** Flèches + AZE

**Synopsis :** (?) On contrôle un vaisseau/personnage et on doit faire face à des vagues d’ennemis et des boss. Le genre de trucs où on a qu’une vie et dès qu’on perds on recommence et le but c’est d’avoir le plus gros score. Plus on va loin, plus le score monte vite. Le joueur a des shields de différent types (genre 2 ou 3, différenciables par la couleur?) qui permettent d’absorber différents types de tirs ennemis. (cc Ikaruga) Du coup le jeu tournera autour de comment bien switch les shields au bon moment plutôt que sur le *pur* skill aux shmups.

(idée) Les vagues d’ennemis donnent + de power up pour le perso/vaisseau du joueur tandis que les boss font plutôt monter le score.

(idée complémentaire) Au bout d’un moment y’a plus du tout de vagues d’ennemis mais que des boss.

**Intérêts de votre jeu à être en ligne (score ? multijoueur ?) :** score en fonction de l’avancée, possibilité de scoreboard

* Morgan : Idée peut-être pas terminée / pas assez précise

**Type de jeu :** 2D Vue de haut en 1VS1  
**Contrôles :** Right click : déplacer (age of empire like), left click : tir / ***activation*** (voir plus bas)

**Synopsis :** On utilise un pistolet magique qui tire des balles se transformant en trou noir lorsqu’on ***active***, le but étant de faire tomber l’adversaire dans un de ses trous noirs tandis que nous on survit (peut-être un petit système de points)

**Intérêts de votre jeu à être en ligne (score ? multijoueur ?) :** Système de défi avec ELO maybe ?

**Idée d’implémentation d’une IA simple pour les ennemis :**

les ennemis possèdent une *detection range* autour d’eux qui leur permet de “voir” les unités ennemis (nos joueurs), si un des joueurs est dans ce cercle, cette unité ennemie va décider d’une action.

Pour cela, on peut créer une fonction getNearbyObjects(position, detectionRange) qui nous retournera tous les objets proches, on peut ensuite vérifier si l’unité est en surnombre, en sous-nombre etc. pour décider de son comportement : agressif ou défensif (se rapprocher d’une tourelle maybe ?)

Exemple (fonction trouvée sur le net) :

public static List< WorldObject > FindNearbyObjects(Vector3 position, float range) {

Collider[] hitColliders = Physics.OverlapSphere(position, range);

HashSet< int > nearbyObjectIds = new HashSet< int >();

List< WorldObject > nearbyObjects = new List< WorldObject >();

for(int i = 0; i < hitColliders.Length; i++) {

Transform parent = hitColliders[i].transform.parent;

if(parent) {

WorldObject parentObject = parent.GetComponent< WorldObject >();

if(parentObject && !nearbyObjectIds.Contains(parentObject.ObjectId)) {

nearbyObjectIds.Add(parentObject.ObjectId);

nearbyObjects.Add(parentObject);

}

}

}

return nearbyObjects;

}

Source : <http://stormtek.geek.nz/rts_tutorial/part22.php>