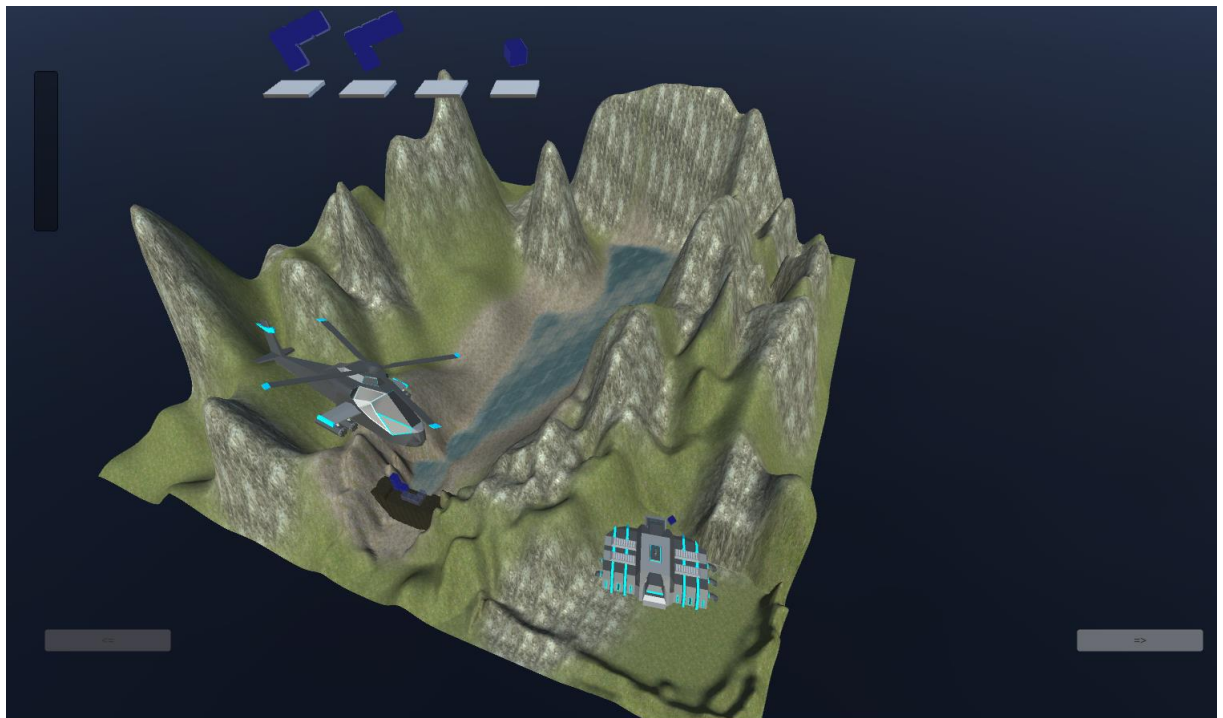


HE-Arc

# Serious Game : Barrain

Travail d'automne



Cyril Ruedin  
27/01/2017

Barrain est un jeu pédagogique créé pour des élèves âgés de 7 à 11 ans. Le but du jeu est de combler le mur d'un barrage en hélicoptérant des pièces depuis une usine de production données jusqu'à la zone de construction et de les poser de manière cohérente. Le pédagogue peut récupérer des données de réalisation du niveau par le joueur pour évaluer ses fonctions exécutives. Il a pour objectif d'être intégré à une plateforme contenant d'autres jeux sérieux pédagogiques.

Le nom « Barrain » est un mix en « Barrage » et « Brain », ce qui exprime la dualité entre la complexité de l'assemblage du barrage et l'utilisation des ressources cognitives du joueur.



Barrain is an educational game created for students aged 7 to 11 years. The goal of the game is to fill the wall of a barrage by heliporting some pieces from a given production. The pedagogue can retrieve the player's performance data to evaluate its executive functions. It aims to be integrated into a platform containing other serious educational games.

The name "Barrain" is a mix in "Barrage" and "Brain", which expresses the duality between the complexity of the assembly of the barrage and the use of player's cognitive resources.

## Table des matières

Table des matières .....	2
Figures .....	4
1. Introduction.....	5
2. Contexte et besoin .....	5
2.1. Serious game pour enfant .....	5
2.2. Besoin d'outils d'évaluation métier.....	5
2.3. Plateforme.....	5
2.4. Objectifs du projet.....	5
3. Principe du jeu.....	7
3.1. Phase de jeu .....	7
3.2. Choix du design .....	7
4. Contexte pédagogique .....	8
4.1. Fonctions exécutives .....	8
4.2. Données extraites.....	8
5. Décisions, choix techniques et pédagogiques.....	8
5.1. Décision en cours de développement .....	8
5.1.1. Hélicoptère .....	8
5.1.2. Cube chanfreiné .....	8
5.2. Entretien avec l'expert pédagogue .....	9
5.2.1. Distance usine entre l'usine et le barrage.....	9
5.2.2. Jeu de pièce de l'usine.....	9
5.2.3. Qualité de construction.....	9
5.2.4. Design .....	9
5.1. Problèmes rencontrés .....	9
5.1.1. Pièces qui volent dans l'espace .....	10
5.1.2. Transfert de pièces dans l'inventaire .....	10
5.2. Choix de design.....	10
5.2.1. Corde entre l'hélicoptère et la pièce.....	10
5.2.2. Cohérence de l'environnement.....	10
5.2.3. Niveau de l'eau dépendant du barrage.....	10
5.2.4. Gestion des contreforts.....	10
6. Partie technique .....	11
6.1. Environnement de développement .....	11
6.2. Objets du jeu .....	11
6.2.1. Barrage .....	11

6.2.2.	Usines, décor et hélicoptère.....	11
6.2.3.	Terrain .....	11
6.2.4.	Pièces.....	11
6.2.5.	Eau .....	11
6.2.6.	Caméra principale.....	11
6.2.7.	Caméra d'inventaire .....	11
6.2.8.	Interface utilisateur .....	11
6.3.	Scripts .....	11
6.3.1.	GameManager .....	12
6.3.2.	Barrage .....	13
6.3.3.	Helicopter .....	13
6.3.4.	Factory.....	13
6.3.5.	Camera .....	13
6.4.	Classes .....	13
6.4.1.	Piece .....	13
6.4.2.	PlayerBarrageStatistique .....	14
6.5.	Définition des niveaux.....	14
6.5.1.	Structure.....	14
6.6.	Intégration dans la plateforme.....	14
7.	Amélioration et développement futur .....	14
7.1.	Préférence utilisateur.....	14
7.2.	Animation lors du lancement d'un niveau .....	14
7.3.	Déplacements plus souples de l'hélicoptère.....	14
7.4.	Aide en cours de jeu .....	15
7.5.	Qualité du code .....	15
7.6.	Amélioration de l'interface utilisateur .....	15
7.7.	Bugs .....	15
7.7.1.	Sélection de pièces par la souris .....	15
7.8.	Tests utilisateurs.....	15
7.9.	Améliorations techniques et pédagogique .....	15
7.9.1.	Génération de niveau XML.....	15
7.9.2.	Création de pièces personnalisées .....	15
8.	Conclusion .....	15
9.	Bibliographie/sources.....	16
10.	Remerciements .....	16
11.	Annexes .....	16

Lieu, date, signature(s) .....	16
--------------------------------	----

## Figures

Figure 1 Structure d'un barrage .....	7
Figure 2 Pièce en forme de L .....	7
Figure 3 Barrage avec cubes chanfreinés et avec cubes simples .....	9
Figure 4 Diagramme des phases de jeu.....	12

## 1. Introduction

Le projet s'inscrit dans le cadre de cinq serious games<sup>1</sup> à but pédagogique pour une tranche d'âge de 7 à 11 ans, déjà développés et « fonctionnels » qu'il faut idéalement regrouper en une plateforme et/ou mettre à niveau. De nombreux aspects peuvent être améliorés, aussi bien dans le portage d'un des serious games sous Unity<sup>2</sup> ou dans l'élaboration de nombreux niveaux et récompenses par exemple.

La proposition a été de créer un nouveau jeu qui s'inspire d'un jeu déjà existant. L'objectif de ce jeu était de copier une pièce en posant des cubes. L'objectif du nouveau jeu est de poser des pièces pour combler une structure.

Le projet est mandaté par la HEP Vaud et est représenté par un expert en pédagogie.

## 2. Contexte et besoin

Le jeu doit être amusant et doit permettre au joueur de développer ses fonctions exécutives. Il doit également fournir des outils permettant aux pédagogues d'évaluer le joueur.

### 2.1. Serious game pour enfant

Le jeu s'adressant à des enfants, il doit être simple à prendre en main et d'une difficulté adaptée. Le réalisme n'est pas très important.

### 2.2. Besoin d'outils d'évaluation métier

Le pédagogue doit pouvoir extraire des informations sur la réalisation du niveau afin d'évaluer, si possible, les fonctions exécutives suivantes :

- **Organisation/planification** : capacité à utiliser des stratégies efficaces, établir des priorités, anticiper et prévoir les étapes d'une tâche.
- **Inhibition** : capacité à résister aux distractions ou à inhiber une réponse attendue ou un commentaire qui nous traverse l'esprit. Cette capacité est souvent comparée à un filtre ou un frein.
- **Flexibilité mentale** : capacité à s'adapter à la nouveauté et aux changements.
- **Jugement** : capacité à évaluer la meilleure alternative face à un problème en fonction des buts à atteindre, des valeurs et des règles sociales. Ceci permet de prendre des décisions appropriées et d'adopter des comportements adaptés aux situations.
- **Autocritique** : capacité à évaluer convenablement ses propres capacités et comportements et à être conscient de ses forces et ses difficultés.

### 2.3. Plateforme

Une plateforme regroupant les jeux est en développement par un étudiant en master. Les jeux doivent être centralisés sur celle-ci.

### 2.4. Objectifs du projet

Les objectifs du projet sont définis par le cahier des charges.

Objectifs primaires :

---

<sup>1</sup> Serious game ou jeux sérieux en français est un logiciel qui combine une intention sérieuse avec des ressorts ludiques

<sup>2</sup> Unity est un moteur de jeu

- Le joueur peut choisir ses pièces et les poser.
- Une interface graphique est prévue (relativement rudimentaire toutefois).
- Une gestion des niveaux sera effectuée via fichier XML.
- Le système de score est géré principalement avec le temps consommé.
- L'eau est gérée par un niveau progressif qui monte au fil du temps.
- Trois niveaux seront disponibles.

Objectifs secondaires :

- Le joueur peut assembler des pièces à l'usine, pour en construire des plus grandes ;
- Amélioration du *scoring* en tenant compte de plusieurs autres paramètres ;
- *Tuning* du système de score ;
- Au moins 10 niveaux ;
- Amélioration de l'interface graphique ;
- Gestion des contreforts ;
- Tutoriel et aide dans les différents niveaux.

### 3. Principe du jeu

Le but du jeu est de combler des « vides » dans une structure de barrage incomplète. Le barrage forme un mur bloquant l'eau du creux entre deux montagnes.



Figure 1 Structure d'un barrage

Pour cela, le joueur dispose de pièces qu'il doit placer dans le barrage. Les pièces peuvent être déplacée et tournée dans toutes les directions.

Une pièce est un assemblage de cube.

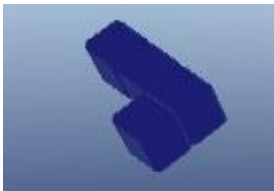


Figure 2 Pièce en forme de L

Les pièces étant produite dans une usine située à proximité du barrage, un hélicoptère sera chargé de transporter les pièces de l'usine au barrage. L'hélicoptère peut embarquer un certain nombre de pièces (définie par le niveau).

#### 3.1. Phase de jeu

Le joueur devra procéder de la manière suivante

- Observer la structure du barrage
- Choisir les pièces dans l'usine et les charger dans l'hélicoptère
- Donner l'ordre à l'hélicoptère de rejoindre le barrage
- Poser les pièces dans le barrage
- Si le barrage est incomplet, retourner à l'usine.

#### 3.2. Choix du design

Le design doit être à la fois simple et efficace. Le joueur doit pouvoir discerner facilement la structure des pièces et du barrage.



## 4. Contexte pédagogique

Afin d'analyser les résultats, le jeu doit permettre de ressortir des statistiques adaptées permettant aux pédagogues d'évaluer les fonctions exécutives du joueur.

### 4.1. Fonctions exécutives

Selon l'expert pédagogue, le jeu fait travailler plusieurs fonctions exécutives :

- Planification, par le choix des pièces
- Récupération active d'information dans la mémoire, lorsque l'usine n'est pas visible en même temps que le barrage. Il faut donc imaginer le barrage.
- Mise à jour des connaissances, avec les différents niveaux de nouveaux problèmes apparaissent.
- Observation et vision en 3D
- Différenciation des objets 3D
- Dextérité manuelle, manipuler des objets dans un environnement virtuelle

### 4.2. Données extraites

Les données extraites permettent d'établir un profil.

Réussir ou échouer un niveau enregistre les données suivantes :

- Nom du niveau
- Réussite
- Temps
- Qualité de construction (Nombre de bloc manquant)
- Nombre de trajet de l'hélicoptère
- Nombre de pièces posées
- Taille moyenne des pièces

## 5. Décisions, choix techniques et pédagogiques

Ce chapitre traite des décisions prises pendant le développement. Elles peuvent aller à l'encontre du cahier des charges

### 5.1. Décision en cours de développement

Ces décisions ont été prises en analysant les contraintes techniques de plus près ou en observant le rendu.

#### 5.1.1. Hélicoptère

Le choix du moyen de transport a été faites selon des contraintes techniques. Il aurait été très difficile de charger des pièces dans un camion et de le faire suivre une route. Un hélicoptère peut se déplacer et se diriger dans tous les sens.

#### 5.1.2. Cube chanfreiné

Les différentes couches des barrages sont difficiles à différencier, les cubes étant collé. Une autre solution aurait été d'utiliser une texture marquant les bords.



Figure 3 Barrage avec cubes chanfreinés et avec cubes simples

## 5.2. Entretien avec l'expert pédagogue

Un entretien a eu lieu le 22 décembre 2016, certaines décisions ont été prises.

### 5.2.1. Distance usine entre l'usine et le barrage

La distance entre l'usine et le barrage est petite pour les premiers niveaux (visibilité simultanée de l'usine et du barrage) et grande pour les niveaux de complexité élevé. Ceci permet d'entraîner la capacité de mémorisation du joueur.

### 5.2.2. Jeu de pièce de l'usine

L'usine propose un jeu de pièce limité à quatre pièces au lieu de proposer toutes les pièces simultanément. Une nouvelle pièce apparaît quand une pièce est hissée dans l'hélicoptère. Le joueur doit donc réagir en fonction des pièces nouvellement apparues et n'est pas submergé par une quantité importante de pièces.

### 5.2.3. Qualité de construction

Le joueur doit remplir tous les blocs manquant, or il n'est pas possible de placer un bloc en dessous d'un autre. Il est donc possible de faire une erreur de construction. La décision a été prise de tolérer un certain nombre d'erreur dépendant du niveau.

Si le joueur commet trop d'erreur, le barrage est détruit et le niveau est échoué.

### 5.2.4. Design

Le design est suffisant à l'état. Le vide entre l'usine et le barrage convient pour le public cible. L'absence de source et de bassin d'écoulement n'est pas problématique non plus. Il n'est donc pas nécessaire de refaire le terrain.

## 5.1. Problèmes rencontrés

Certains problèmes techniques ont été rencontré, et ont pu être résolu en utilisant des astuces.

#### 5.1.1. Pièces qui volent dans l'espace

Il arrivait que des pièces disponibles à l'usine subissent des forces dues aux mouvements d'autres pièces qui les envoient hors de la zone de jeu. Une cage invisible est construite autour pour que les pièces restent dans un périmètre restreint.

#### 5.1.2. Transfert de pièces dans l'inventaire

Les pièces déplacées ont tendance à se démembrer lors du transfert vers la zone d'inventaire, la mise à jour des jonctions avec les autres cubes s'avère problématique et cas de téléportation. Ce phénomène est devenu trop important lors du déplacement usine-inventaire. La reconstruction de la pièce pouvant prendre jusqu'à quelques minutes. Le problème est résolu en supprimant les composants de la pièce puis en les réinstanciant directement à la position de destination.

### 5.2. Choix de design

Les choix de design ont été faits pour prioriser le développement du Game Play.

#### 5.2.1. Corde entre l'hélicoptère et la pièce

Créer une corde entre deux objets est très complexe. Dans la fiction, les soucoupes volant enlèvent des objets avec un spot très lumineux. Le concept a été repris, et est très simple et efficace. Et plus de fonctionner correctement, le jeu reste dans un thème futuriste et l'animation est belle à voir.

#### 5.2.2. Cohérence de l'environnement

Un barrage est normalement constitué d'une source, d'un lac et d'un bassin d'écoulement. Il était initialement prévu de corriger les terrains afin que le jeu devienne plus réaliste. Corriger l'environnement aurait été très compliqué, car il aurait fallu gérer des courants d'eau, et cela aurait pris beaucoup de temps.

Toutefois, selon l'expert, cela n'a aucune importance pour des enfants cibles. Le design convenait, il n'était donc pas optimal de passer beaucoup de temps là-dessus. Un jeu vidéo n'a pas forcément lieu d'être réaliste, la vie de tous les jours nous fait déjà suffisamment profiter des contraintes de la réalité.

#### 5.2.3. Niveau de l'eau dépendant du barrage

Il était initialement prévu que l'eau monte progressivement au fil du temps. Il est très difficile de simuler un écoulement de l'eau en cas de débordement.

L'eau dépend donc de la construction du barrage, elle monte quand une couche est terminée. Cette solution a également l'avantage de récompenser le joueur quand il a terminé une couche, un peu comme la disparition d'une ligne dans un Tetris.

#### 5.2.4. Gestion des contreforts

Les contreforts sont indispensables dans les barrages non voûtés. Cependant, l'intégration dans le jeu a été abandonnée car ils n'apportent pas une utilité réelle pour le public cible et est compliqué à implémenter.

## 6. Partie technique

Ce chapitre traite de la conception technique, de la logique de jeu et de la constitution des différents éléments.

### 6.1. Environnement de développement

Le jeu a été développé avec le moteur de jeu Unity 5.4. Les scripts sont codés en C# avec Visual Studio 2015.

### 6.2. Objets du jeu

Liste des objets physiques du jeu, visibles par l'utilisateur. Ce sont des « GameObject » dans Unity.

#### 6.2.1. Barrage

Tout comme les pièces, le barrage est composé de cubes chanfreinés. Seule la couleur est différente. Les cubes dépendent du niveau et sont immobile une fois instanciés.

#### 6.2.2. Usines, décor et hélicoptère

Téléchargé depuis l'« Asset Store » de Unity. Disponible dans le pack « Warzone Environment pack ». Le rotor de l'hélicoptère est animé.

L'usine est placée à deux endroits, une est proche du barrage est l'autre est plus éloignée. L'usine activée dépend du niveau.

#### 6.2.3. Terrain

Les deux terrains (Usine et barrage) ont été dessinés manuellement avec l'outil « Terrain » de Unity. La tâche consiste à tracer le relief et la texture.

#### 6.2.4. Pièces

Les pièces sont formées de cubes chanfreinés collés. Les cubes chanfreinés ont été importés depuis Blender au format OBJ. Il n'a pas été possible d'appliquer des textures, car les coordonnées de textures sont manquantes. Cependant les couleurs uniformes donnent un rendu moderne et élégant.

#### 6.2.5. Eau

L'eau provient des Assets standard de Unity. Elle s'utilise de la même manière qu'un objet standard. Elle monte et descend selon le niveau de construction du barrage.

#### 6.2.6. Caméra principale

La caméra principale affiche les différents éléments, elle peut être déplacée.

#### 6.2.7. Caméra d'inventaire

La caméra d'inventaire est fixe, elle permet d'afficher les pièces contenues dans l'hélicoptère.

#### 6.2.8. Interface utilisateur

L'interface utilisateur est composé de deux canvas :

- Canvas de gestion, composé d'un dropdown de choix de niveau, d'un bouton pour charger le niveau et un bouton pour quitter l'application.
- Canvas de jeu, composé des boutons pour diriger l'hélicoptère vers le barrage ou vers l'usine

### 6.3. Scripts

Les scripts sont liés à des « GameObject ». Ils établissent les mécaniques de jeu.

### 6.3.1. GameManager

Le script GameManager est lié à la camera principale, car elle est toujours instanciée. Il implémente plusieurs fonctionnalités nécessaires au déroulement du jeu :

- Chargement du niveau, avec la gestion du dropdown
- Gestion de l'UI, activation et désactivation des différents éléments
- Ordre de déplacement de l'hélicoptère par les boutons
- Gestion de stock de l'hélicoptère, sélection de la pièce à poser sur le barrage (clic)
- Enregistrement des statistiques utilisateurs
- Activation de l'usine

Le déroulement de la partie est géré avec une énumération « GameState » qui contient les états suivant :

- Start : Chargement du niveau, début de la partie.
- Factory : Etape de sélection de la pièce dans l'usine. Hélicoptère situé sur l'usine.
- Barrage : Etape de sélection de la pièce dans le stock de l'hélicoptère. Hélicoptère situé sur le barrage.
- Pose : Etape de pose de la pièce sur le barrage.
- Transport : Laps de temps entre deux états, vol de l'hélicoptère, le joueur n'a rien à faire.
- Terminated : Niveau terminé, survient quand le barrage est complet, quand la jauge du carburant est vide ou quand la qualité de construction du barrage est trop faible

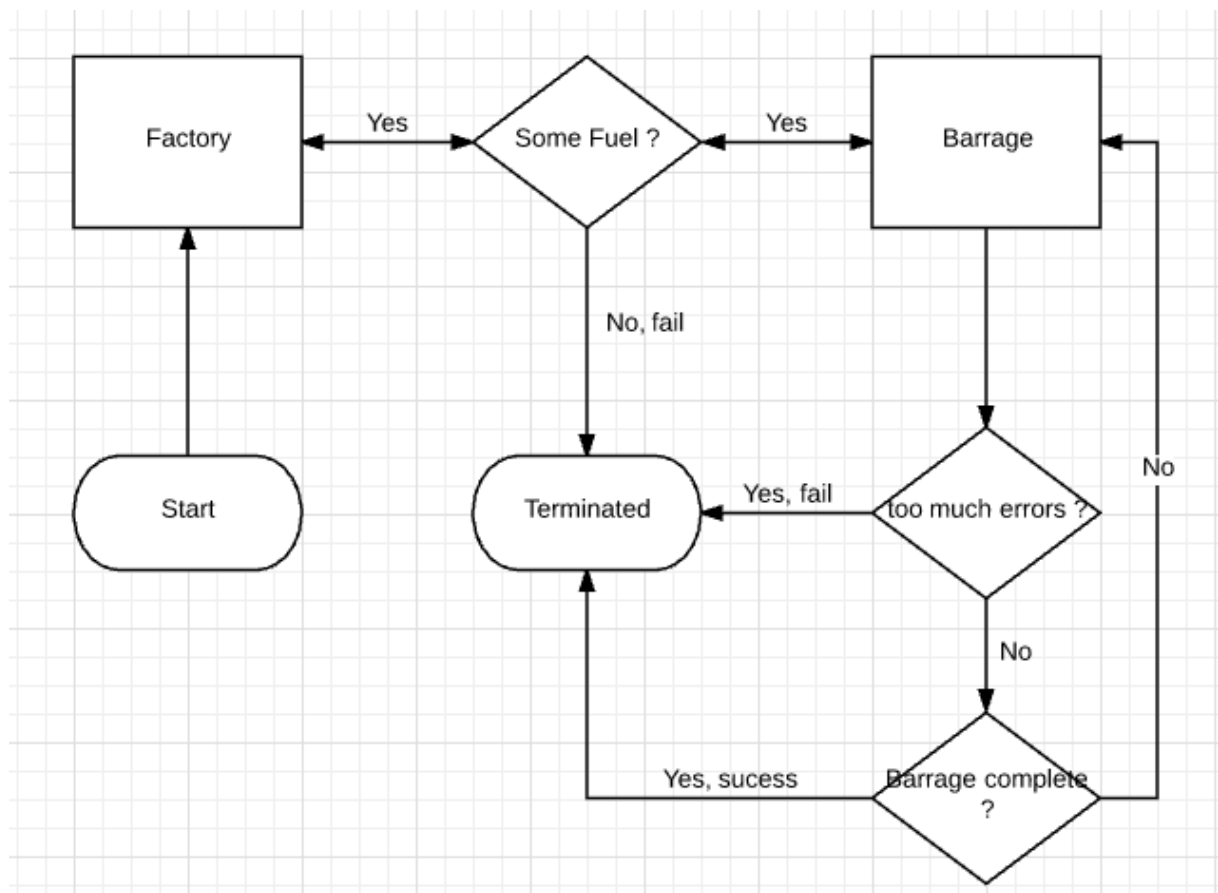


Figure 4 Diagramme des phases de jeu

### 6.3.2. Barrage

Le script barrage est lié à un bloc invisible posé à l'emplacement du barrage. Ce bloc sert comme point de repère à l'hélicoptère.

Il est composé d'un tableau en 3 dimensions représentant la structure du barrage. Une case de cette matrice est liée à un cube du barrage, la valeur est définie si le cube est comblé ou est vide

Le script gère les mécaniques de jeu liées au barrage et la gestion de la matrice :

- Capture des événements clavier et traitement des frappes
- Génération de la structure du barrage
- Placement des pièces
- Gestion de l'eau (monte jusqu'à niveau de la plus haute couche complète du barrage).
- Destruction du barrage en cas d'échec
- Détection des erreurs, ajout d'un bloc rouge pour les combler
- Transparence du barrage en cas d'erreur

### 6.3.3. Hélicoptère

Le script hélicoptère est lié au « GameObject » hélicoptère. Il gère les fonctionnalités de l'hélicoptère :

- Déplacement jusqu'à un « GameObject », gestion de la direction
- Gestion du carburant et affichage de la jauge
- Chargement d'une pièce
- Allumage et extinction du spot lors du chargement des pièces

### 6.3.4. Factory

Le script factory est lié à l'usine.

- Instanciation et construction des pièces à disposition
- Génération des pièces
- Détection du clic pour sélectionner une pièce

### 6.3.5. Camera

Les scripts de caméra sont liés à la caméra principale, ils servent à contrôler la caméra. Ils ont été téléchargés depuis l'« Asset Store » et ont été légèrement adaptés.


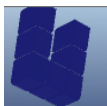
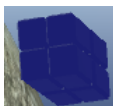
- Déplacer la caméra : maintenir clic droit et bouger la souris
- Rotation de la caméra : maintenir clic gauche et bouger la souris
- Avancer, reculer la caméra : molette avant, arrière

## 6.4. Classes

Contrairement aux scripts, les classes ne sont pas liées à des objets.

### 6.4.1. Piece

La classe définit une pièce par une liste de vecteur représentant la position des cubes. La pièce peut être construite par des modèles prédéfinis, identifiés par une chaîne de caractère :

- L :  U :  Cube2 : 
- T, Cross, Cube1, Rect2x2x1

La pièce possède des méthodes permettant d'instancier et de détruire tous ses blocs sous forme de « GameObject ». Elle conserve également une référence sur chaque cube instancié, ayant une dualité logique et physique.

La classe possède des méthodes de rotation, de recherche des points les plus bas dans la pièce. Cette méthode est notamment utile dans la pose de pièce du barrage.

Elle a également une méthode statique permettant de rechercher à quelle pièce appartient un bloc, utile notamment pour la sélection de pièce à partir d'un clic, Unity permettant de retrouver le « GameObject » cliqué.

#### 6.4.2. PlayerBarrageStatistique

La classe d'enregistrement des statistiques du joueur permet de contenir les différentes informations des résolutions de niveau. Elle possède une méthode pour se sérialiser dans un fichier binaire.

Elle est construite de la même manière que les autres classes d'enregistrement de statistique de la plateforme de serious game.

Un outil permettant d'obtenir de transformer ce fichier en csv est fourni. En attendant que la plateforme supporte pleinement le jeu, il permet lire les résultats avec un tableur.

### 6.5. Définition des niveaux

Les niveaux sont définis dans un fichier XML qui doit se trouver dans le dossier Level. Le nom du niveau est le nom du fichier XML.

#### 6.5.1. Structure

- Epaisseur et hauteur du barrage
- Capacité de l'hélicoptère
- Capacité du réservoir de l'hélicoptère
- Nombre d'erreurs de construction tolérées
- Distance usine-barrage, proche ou éloignée
- Liste des pièces disponibles
- Structure initiale du barrage

### 6.6. Intégration dans la plateforme

Le jeu est fait pour être facilement intégrable. Il suffit de copier le dossier Asset. Les extractions de résultats sont normalisées avec les autres jeux.

## 7. Amélioration et développement futur

Si le jeu devait être repris et amélioré, les points suivants devraient être traités.

### 7.1. Préférence utilisateur

Un enregistrement des niveaux déjà terminés pourrait améliorer la reprise du jeu. Afin que l'ajout de niveau devienne plus agréable, l'emplacement des fichiers des niveaux devrait pouvoir être modifié.

### 7.2. Animation lors du lancement d'un niveau

Un niveau pourrait être introduit avec une vue sur le barrage, puis une transition sur l'usine.

### 7.3. Déplacements plus souples de l'hélicoptère

Les déplacements de l'hélicoptères sont parfois un peu hasardeux, l'algorithme demande à être amélioré.

#### 7.4. Aide en cours de jeu

Un didacticiel pourrait simplifier la prise en main lors des premiers niveaux. Il serait composé d'explications à chaque transition d'état du jeu et d'un rappel des touches.

#### 7.5. Qualité du code

Pour cause de manque d'expérience initial avec Unity, le code est mal organisé. Afin d'augmenter la maintenabilité, il devrait être revu et mieux documenté.

#### 7.6. Amélioration de l'interface utilisateur

Les jauges de carburant et de qualité du barrage sont difficiles à remettre dans leur contexte. Rien n'indique à quoi elles servent. Une amélioration du design est nécessaire, et un effet de clignotement lorsque la jauge atteint ses limites avvertirait efficacement l'utilisateur de la situation.

Des boutons rejouer et niveau suivant seraient appréciables.

#### 7.7. Bugs

Certains bugs sont toujours présents, il est probable que d'autres n'aient pas été répertoriés.

##### 7.7.1. Sélection de pièces par la souris

Il faut parfois cliquer plusieurs fois pour que la pièce se sélectionne. Le problème n'est pas corrigé, mais le jeu reste toutefois parfaitement jouable.

#### 7.8. Tests utilisateurs

Le jeu sera prochainement testé en classe par des enfants. Il a déjà été testé par l'expert pédagogue, selon elle le jeu répondrait tout à fait aux objectifs.

#### 7.9. Améliorations techniques et pédagogique

Cette partie améliorerait la tâche des pédagogues.

##### 7.9.1. Génération de niveau XML

La création d'un niveau manuellement est fastidieuse, créer un outil pour en générer ne sera pas très compliqué et réduirait considérablement l'effort.

Des informations sur le nom de niveau, de sa difficulté pourrait être ajouté dans le fichier.

##### 7.9.2. Création de pièces personnalisées

Le jeu ne permet que d'utiliser des pièces prédéfinies. Des pièces personnalisées apporterai plus de dimension à la jouabilité. Toutefois, cela augmenterait aussi la complexité, ce qui ne serait pas forcément adapté à des enfants.

## 8. Conclusion

Malgré l'interface utilisateur modeste, le jeu est parfaitement exploitable. Le jeu semble amusant et des fonctions cognitives sont mises à l'épreuve. Les statistiques utilisateurs sont automatiquement enregistrée quand le joueur quitte le jeu à l'aide du bouton intégré.

Les objectifs ont été globalement atteints ou adaptés. Certains ont été dévalués aux cours du projets car ils ne correspondaient pas au besoin réel, par exemple la gestion des contreforts.

Il reste encore à ajouter plusieurs niveaux de jeu dans les livrables.

De nouvelles perspectives s'ouvriront une fois les tests en classe effectués.



## 9. Bibliographie/sources

- Unity Asset Store
- Tutoriel et forum unity3d.com
- Msdn Microsoft C#
- Divers forums (stackoverflow, developpez.com)

## 10. Remerciements

Certaines personnes ont été indispensables à la réussite du projet :

- Florence Quinche, l'expert pédagogique, pour ses compétences et son suivi
- Julien Marchand, le superviseur, pour ses conseils avisés
- L'équipe des séances synergies, pour leur soutien tout au long du projet

## 11. Annexes

Les annexes sont disponibles sur le répertoire de rendu et sur la forge.

- Documentation utilisateurs
  - Manuel de configuration
  - Joueur
- Code source et fichiers Unity.
  - Adresse de la forge : [https://github.com/cyrilruedin/SG\\_Barrage](https://github.com/cyrilruedin/SG_Barrage)
- Exécutable Windows et Mac
- Cahier des charges
- PV entretien avec l'expert en pédagogie

Lieu, date, signature(s)