# **Blob Runner - Documentation Technique**

## **Introduction**

Ce document décrit la structure et le fonctionnement du projet Blob Runner, développé sous Unity. Il détaille chaque script, son rôle, les interactions entre les composants, ainsi que les choix techniques effectués.

## **1. Architecture Générale**

* **Gestion du joueur** : BlobController, JumpBlob, BiomassManager
* **Collectibles** : CollectibleBiomass
* **Obstacles** : HorizontalLaser, LaserGrid, PanneauCommande, PlaqueElectrique, PyloneObstacle, PorteAutomatique
* **Progression du niveau** : ChunkManager
* **Interface utilisateur** : GameHUD
* **Gestion globale du jeu** : GameManager
* **Mécaniques spéciales** : TubeLangue

## **2. Gestion du joueur et biomasse**

### **2.1 BlobController**

* Gère le comportement de base du blob
* Interagit avec JumpBlob pour le saut sol/plafond et BiomassManager pour la biomasse

### **2.2 JumpBlob**

* Permet au blob de sauter entre sol et plafond
* Animation fluide avec DOTween
* La variable isCeiling indique la position actuelle, utilisé pour lasers et TubeLangue

### **2.3 BiomassManager & CollectibleBiomass**

* BiomassManager gère la biomasse du joueur
* CollectibleBiomass augmente la biomasse avec effets visuels et sonores
* Les collisions avec obstacles retirent de la biomasse et peuvent déclencher le Game Over

**Choix techniques** :

* Séparation logique joueur / collectibles → maintenance facile
* Animation et physique séparées → gameplay fluide

## **3. Interface utilisateur (HUD)**

### **GameHUD**

* Affiche :
  + Biomasse avec couleurs dynamiques (rouge/jaune/vert)
  + Forme du blob selon biomasse
  + Distance parcourue
  + Collectibles
* Change visuellement la forme du blob selon la biomasse

**Choix techniques** :

* Indépendant du gameplay → modification simple
* Feedback visuel immédiat pour le joueur

## **4. Gestion des niveaux**

### **ChunkManager**

* Génère des sections de niveau procéduralement
* Utilise une queue pour gérer les chunks visibles et supprimer les anciens

**Avantages** :

* Procédural → extensible et infini
* Queue → optimisation mémoire

## **5. Obstacles et interactions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Obstacle** | **Type** | **Interaction** | **Esquive possible** |
| HorizontalLaser | Laser sol/plafond | BiomassManager | Oui, changer de plan |
| LaserGrid | Grille verticale | BiomassManager | Seulement en mode flaque |
| PlaqueElectrique | Plaque au sol | TakeObstacleDamage + ralentissement | Non |
| PyloneObstacle | Pylône | TakeObstacleDamage + ralentissement | Non |
| PorteAutomatique | Porte | Activation via PanneauCommande ou TubeLangue | N/A |
| PanneauCommande | Panneau | Active portes liées | N/A |

**Choix techniques** :

* Colliders + OnTrigger → gestion simple des obstacles
* Interaction avec BiomassManager pour les dégâts → centralisation

## **6. TubeLangue**

* Mécanique spéciale pour interaction à distance
* Gestion complète : extension, rétraction, cooldown, détection collision pendant l’extension

### **Interactions possibles**

|  |  |
| --- | --- |
| **Objet touché** | **Effet TubeLangue** |
| PanneauCommande | Active toutes les portes liées |
| HorizontalLaser | Désactive le laser |
| LaserGrid | Désactive la grille |
| CollectibleBiomass | Ramasse et ajoute biomasse |
| PorteAutomatique | Ouvre la porte si fermée |

**Choix techniques** :

* Séparation visuel (LineRenderer) / logique (interactions)
* Permet gameplay varié sans complexité des contrôles

## **7. Gestion globale du jeu**

### **GameManager**

* Centralise : biomasse, collectibles, distance parcourue
* Met à jour GameHUD
* Gère Game Over et redémarrage

**Choix techniques** :

* Point central de gestion → cohérence des stats
* Simplifie le contrôle des systèmes

## **8. Conclusion et bonnes pratiques**

* Modularité : chaque script a un rôle précis
* Séparation logique / visuelle : ex. TubeLangue
* Feedback joueur immédiat : HUD dynamique, sons, particules
* Gameplay dynamique : obstacles variés, changements de forme, saut sol/plafond, interactions à distance
* Réutilisabilité : composants indépendants facilement déplaçables