# **Proposition de variante du jouet Dynamo**

Le joueur contrôle un personnage à la troisième personne dans un environnement 3D qui a la particularité d’avoir une gravité zéro. En fonction des actions du joueur des géométries apparaissent dans l’environnement de manière procédurale (voir document d’Enki).

## 

## **Déplacement du personnage :**

Pour déplacer son personnage le joueur doit être collé à une surface pour pouvoir se propulser. Le personnage se propulse dans la direction dans laquelle le joueur est en train de viser (à l’aide d’un viseur au centre de l’écran), comme une sorte de « wall jump » mais sans gravité.

Étant dans un environnement Zéro-g le personnage ne peut pas tomber, par contre il est sujet à la direction qu’il a emprunté jusqu’au moment où il arrive à une nouvelle surface qui lui permettra d’emprunté une nouvelle direction.

Pour faciliter les manoeuvres le joueur peut faire straffer légèrement le personnage dans n’importe quelle direction grâce aux touches ZQSD, même quand il n’est pas accroché à une surface.

## **Génération procédural :**

La génération procédurale marche pour la plus grande partie comme le décrit le document d’Enki, il y a juste le straffe qui remplace le saut et les objets au lieu de disparaitre ils sont permanents.

## 

## 

## **Formalisation des propriétés systémiques du jouet :**

L’avatar possède les propriétés suivantes :

* Collision
* Taille
* Position

o Direction

* Orientation caméra

Les géométries possèdent les propriétés suivantes :

* Existence

o Position

o Collision

o Taille

o Visibilité

Pour l’instant le joueur à la main que sur l’existence des objets en fonction des actions de l’avatar. On pourrait aussi imaginer que le joueur ai de l’influence sur d’autres propriétés des objets comme la position, la collision, la taille, la visibilité.

**Exploration des différents branchements possibles entre les propriétés de l’avatar et les objets :**

**Collision (Avatar) <> Position (Objets)** : quand le joueur collisione un objet cet objet commence à bouger dans la direction d’où il a été percuté. Le joueur pourrait encore se propulser à partir de cet objet, ce qui lui impose une direction contraire à la direction dans laquelle le joueur visait.

* A partir de là on peut aussi imaginer un branchement **Collision (Objet) <> Position (Objet)** ou les objets pourraient se percuter entre eux et potentiellement commencer une réaction en chaine.

**Collision (Avatar) <> Collision (Objets) :** quand l’avatar impacte un objet cet objet reste collé à lui (Katamari en Zéro-G quoi).

**Collision (Avatar) <> Taille (Objet) :** quand l’avatar impacte un objet celui-ci devient plus petit ou plus grand.

**Collision (Avatar) <> Existence (Objet) :** quand l’avatar impacte un objet celui-ci disparait.

**Collision (Avatar) <> Visibilité (Objet) :** quand l’avatar impacte un objet celui-ci devient invisible.

**Orientation caméra** **(Avatar) <> Visibilité (Objets) :** le joueur pourrai viser un objet et le rendre invisible grâce à un input dédié. Ca pourrait être intéressant dans le cas où le joueur ai à piéger des ennemies, soit IA soit autres joueurs.

**Orientation caméra** **(Avatar) <> Existence (Objets) :** le joueur pourrai viser un objet et le rendre détruire grâce à un input dédié.

**Orientation caméra (Avatar) <> Position (Objets) :** le joueur pourrai viser un objet et le « prendre » pour ensuite le déplacer à sa guise avec un input dédié.

**Orientation caméra (Avatar) <> Collision (Objets) :** le joueur pourrai viser un objet et lui enlever sa collision avec un input dédié pour passer à travers, en perdent la visibilité de l’environnement pendant un moment. Ceci peut permettre au joueur de conserver sa trajectoire en perdent de l’information momentanément.