# CoopBot

## Concept :

Deux équipes doivent coopérer pour explorer un niveau et récupérer des objets dispersés sur la carte. Pour forcer la coopération, chacune des deux équipes n'aura accès qu'à une partie de la carte. Les unités accessibles dans chacune des deux équipes seront différentes et des obstacles demandant l'intervention d'une unité pour surmonter l'obstacle est obligatoire. Dans certain cas, des obstacles bloquant l'équipe 1 ne pourront être détruit que par l'équipe 2, obligeant alors les deux équipes à communiquer pour surmonter l'obstacle en question.

## Public visé :

Tout comme pour WarBot, ce jeu cible des adolescents n'ayant jamais ou peu touché à la programmation (le but étant de les initier à ce domaine).

## Type de jeu :

Le jeu demande de mettre en place des stratégies à plusieurs niveaux pour réussir de finir le jeu :

- Au niveau micro en coordinant les unités au sein d'une même équipe
- Au niveau macro en répondant et demandant l'assistance des unités de l'autre équipe.

Pour cela, il faudra utiliser le système de messages. La communication entre deux équipes se fera via la base : les unités d'une équipe pourront envoyer des messages à chacunes des unités composant son équipe mais également à la base de l'autre équipe.

## Direction artistique :

Le jeu prend place dans une mine. Les unités de l'équipe A sont des nains qui doivent récupérer des pierres précieuses. Les unités de l'équipe B sont des créatures vivant dans la lave de la mine.

## Obstacles :

### Rochers

Les rochers sont des obstacles positionnés sur le territoire de l'équipe A et qui empêche la progression des unités de cette équipe. Pour les détruire, il faut qu'un Soldat Lourd tire en direction de ce rocher.

### Trou

Les trous sont des obstacles positionnés sur le territoire de l'équipe B et qui empêche la progression des unités de cette équipe. Pour résoudre ce problème, un Ingénieur doit construire un pont sur ce trou.

### Ennemis

Les ennemis sont des obstacles pouvant blesser les unités des deux équipes via diverses attaques. Il y en a deux types : ceux sur le territoire de l'équipe A et ceux sur le territoire de l'équipe B. Pour les détruire il faut que les soldats des deux équipes les attaquent.
Pour encourager la coordination, ces ennemis verront leur vie régénérer au fur et à mesure du temps pour éviter que des unités envoyé une à une contre ces ennemis permettent de les tuer.

## Unités équipe A :

### Marchand

Les marchands sont des unités à grande vitesse de déplacement possédant un sac d'une grande taille. Ils sont idéaux pour permettre à l'équipe d'explorer, échanger les ressources entre les deux bases et réapprovisionner les autres unités.

### Soldat léger

Les soldats léger sont des unités à vitesse de déplacement moyenne qui sont capables de tirer sur des ennemis.
Ils ont peu de vie et tir rapidement et ont un sac à grande capacité.

### Ingénieur

Les ingénieurs sont des unités qui permettent de créer des ponts de lave au-dessus des trous qui empêchent la progression des unités de l'équipe B.
Ils ont peu de vie et possède un sac de capacité moyenne.

## Unités équipe B :

### Marchand

Les marchands sont des unités à grande vitesse de déplacement possédant un sac d'une grande taille. Ils sont idéaux pour permettre à l'équipe d'explorer, échanger les ressources entre les deux bases et réapprovisionner les autres unités.

### Soldat lourd

Les soldats lourds sont des unités à vitesse de déplacement faible qui sont capables de cracher de l'acide sur des ennemis. Ils sont également capable de cracher sur des rochers qui bloquent l'avancé de l'équipe A.
Ils ont beaucoup de vie, tir lentement et ont un sac à faible capacité.

### Transporteurs

Les transporteurs sont des unités permettant aux unités de l'équipe A de se déplacer sur le territoire de l'équipe B. Lorsqu'une unité de l'équipe A se retrouve sur la même case qu'une unité de type transporteur, alors seule les actions de l'unité de l'équipe A sont effectués.

# Règles du jeu :

### Territoire

Les cartes de jeu sont découpés en deux parties appelées territoire. Chaque équipe est limité à son territoire. Par conséquent, une unité de l'équipe A ne peut pas se déplacer sur le territoire de l'équipe B.

### Cartes du jeu

Le jeu proposera plusieurs carte dans lesquelles les bases (une par équipe), les obstacles et les objets à collecter ont une position fixe. Ces cartes seront de format 16:9. Les bases doivent se trouver à l'intersection entre les deux territoires des équipes (pour que chaque unité puisse accéder aux deux bases)

### Paramétrage de la partie

Avant de démarrer la partie, les joueurs se mettent d'accord sur quelle carte ils souhaitent jouer ainsi que sur le nombre d'unités qu'ils souhaitent avoir pendant la partie. La modification du nombre d'unités se fait via un slider différent pour chaque type d'unité. Ce slider va de 0 à 10.

### Fin d'une partie

Une partie se termine quand tous les objets à collecter ont été ramassés. Il n'y a pas besoin que les objets soient tous détenu par la même équipe.

### Double inventaire

Chaque unité possède deux inventaire : un pour les ressources produitent par la base A et un deuxième pour les ressources produitent par la base B.

### Ressource

Les bases produisent des ressources au cour du temps. Lors de la création d'une unité, l'inventaire correspondant à son équipe est rempli. Il n'y a pas de ressources récupérables sur la carte.
Les deux bases produisent des ressources différentes.

### Base et échange de ressource

Les unités ne peuvent pas donner à une base les ressources que cette base produit. Il est cependant possible de lui donner les ressources correspondant à l'autre équipe.

### Vie des ennemis

Si un ennemi n'a pas reçu de dégâts depuis un certain temps, sa vie se régénère.

#### Communication

En plus de pouvoir envoyer des messages aux unités de leur équipe, les unités peuvent envoyer des messages à la base de l'autre équipe.

## Liste des primitives :

### Variables

#### Porté tir

Un float valant la porté de tir de l'unité (vitesseProjectile * dureeDeVieProjectile). Cette variable est spécifique aux soldats lourds et léger.

#### Vie actuelle

Donne la vie actuelle de l'unité.

#### Ressources A actuelle

Donne le nombre de ressources produite par la base de l'équipe A de l'unité.

#### Ressources B actuelle

Donne le nombre de ressources produite par la base de l'équipe B de l'unité.

#### Distance cible

Cette primitive donne la distance entre l'unité sa cible.

#### Distance objectif

Cette primitive donne la distance entre l'unité son objectif.

#### Nombre membre

Cette primitive retourne le nombre d'unité dans un groupe. Le groupe dont l'on souhaite compter les membres est spécifié via une chaîne de caractères. Si le groupe n'existe pas, cette primitive retourne 0.

#### Lancé de dé

Cette primitive retourne un nombre aléatoire entre 0 et la borne maximale, défini en paramètre.

#### Constante

Cette primitive permet d'écrire n'importe quelle constante de type chaîne de caractère ou nombre.

## Conditions

### Niveau de vie

Cette primitive retourne vrai si la vie de l'unité est supérieur à un double entre 0 et 1 ( défini en paramètre ) multiplié par la vie maximale de l'unité. Return vieActuelle > vieMax * param

### Niveau de ressource A

Cette primitive retourne vrai si le nombre de ressource produite par la base de l'équipe A de l'unité est supérieur à un double entre 0 et 1 ( défini en paramètre ) multiplié par taille maximale de son inventaire. Return ressourceBActuelle > tailleInventaire * param

### Niveau de ressource B

Cette primitive retourne vrai si le nombre de ressource produite par la base de l'équipe B de l'unité est supérieur à un double entre 0 et 1 ( défini en paramètre ) multiplié par taille maximale de son inventaire. Return ressourceBActuelle > tailleInventaire * param

### Peut tirer

Cette primitive est spécifique aux soldats lourds et léger. Elle retourne vrai si l'unité est à distance de tir de sa cible et si son arme est rechargée.

### Chargé

Cette primitive est spécifique aux soldats lourds et léger. Elle retourne vrai si l'arme de l'unité est rechargée.

### Message reçu

En utilisant cette primitive, on modifie la cible de l'unité. Cette primitive retourne vrai si l'unité a reçu un message provenant de son équipe. Il est possible de mettre en paramètre une chaîne de caractère pour filtrer les messages reçus (on vérifiera alors si l'unité a reçu un message dont le texte est le même que celui passé en paramètre). Si le paramètre est vide, on retourne vrai si on a reçu un message, quelque soit son contenu.

### Message reçu d'un coéquipier

En utilisant cette primitive, on modifie la cible de l'unité. Cette primitive est spécifique aux bases. Cette primitive retourne vrai si l'unité a reçu un message provenant de l'autre équipe. Il est possible de mettre en paramètre une chaîne de caractère pour filtrer les messages reçus (on vérifiera alors si l'unité a reçu un message dont le texte est le même que celui passé en paramètre). Si le paramètre est vide, on retourne vrai si on a reçu un message, quelque soit son contenu.

### Inférieur, supérieur, égal…

Cette primitive prend deux variables en paramètre. Elle retourne vrai si l'opération est correcte. Il est possible de passer d'un test d'égalité, supériorité… en modifiant le signe de la primitive via un menu déroulant.

#### A un objectif

Cette primitive retourne vrai si l'objectif de l'unité n'est pas null.

#### Objectif de type

Cette primitive retourne vrai si l'objectif est d'un certain type d'unité. Elle possède un menu déroulant dans lequel on retrouve tous les types d'objets du mode (message, drapeau, canard…)

#### Proche de

En utilisant cette primitive, on modifie la cible de l'unité. Cette primitive retourne vrai si on est proche d'une unité d'un certain type. Cette primitive possède un menu déroulant dans lequel on retrouve tous les types d'entité possibles (drapeau, baleine, canard…).

#### Dans le groupe

Cette primitive retourne vrai si l'unité appartient au groupe spécifié en paramètre. Ce paramètre est une chaîne de caractère.

#### Peut donner ressource A

Vérifie si l'unité peut donner une ressource produite par l'équipe A à sa cible.

#### Peut donner ressource B

Vérifie si l'unité peut donner une ressource produite par l'équipe B à sa cible.

### Actions terminales

#### Avance

Cette action termine le tour de l'unité. L'unité se déplace en suivant une ligne droite. La distance parcourue dépend de la vitesse de l'unité.

#### Tir

Cette action termine le tour de l'unité et est spécifique aux baleines et aux canards. L'unité tir en direction de sa cible si son arme est chargé. Sinon, la primitive termine simplement le tour. Consomme des ressources de l'équipe de l'unité.

#### Construire

Cette action est spécifique à l'ingénieur. Elle permet de construire un pont sur un trou. Consomme à la fois des ressources A et des ressources B.

#### Embarquer

Cette action est spécifique aux unités de l'équipe A (excepté la base). Elle permet d'utiliser un transporteur.

#### Donner ressource A

Cette action permet de donner à sa cible une ressource produite par l'équipe A.

#### Donner ressource B

Cette action permet de donner à sa cible une ressource produite par l'équipe B.

#### Peut créer

Cette action est spécifique aux bases. Elle possède un menu déroulant pour sélectionner un type d'unité. Vérifie si l'unité peut créer l'unité passé en paramètre.

### Actions non terminales

#### Tourne

Cette primitive prend une variable en paramètre. L'unité se tournera de x%360 degré.

#### Tourne vers

Cette primitive possède un menu déroulant pour choisir entre la cible et l'objectif. L'unité se tourne en direction de ce paramètre.

#### Définir objectif

Cette primitive permet de modifier la valeur de la variable objectif en lui affectant la valeur de la variable cible.

#### Supprimer objectif

Cette primitive permet de supprimer la valeur de l'objectif (objectif = null).

#### Rejoindre groupe

Cette primitive permet de faire rejoindre un groupe à une unité. Elle prend en argument une chaîne de caractère correspondant au groupe que l'on souhaite rejoindre. Si le groupe n'existe pas encore, il est créé lorsque l'unité tente de le rejoindre.

#### Quitter groupe

Cette primitive permet de faire quitter un groupe à une unité. Elle prend en paramètre une chaîne de caractère correspondant au groupe que l'on souhaite quitter.

### Messages

#### Envoi message

Cette primitive permet d'envoyer un message. Elle prend deux argument : le groupe auquel on souhaite envoyer le message et le message en lui même.

#### Répondre message

Cette primitive permet d'envoyer un message à l'expéditeur du message ciblé. Cette primitive prend une chaîne de caractère en argument.

### Transférer message

Cette primitive est spécifique aux bases. Cette primitive permet d’envoyer un message provenant de l’autre équipe aux unités de son équipe. Cette primitive prend une chaîne de caractère en argument.