Game Design Document

PANIC AT THE SCIENCE SHACK

”Sort’Em !”

Robert Biaulet

SOMMAIRE

VUE D'ENSEMBLE

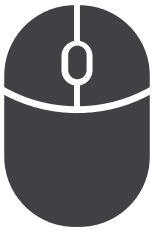
- **Genre :** Jeu d'assortiment et de rapidité
- **Gameplay abstrait :** Il y a, au sein d'un espace fixe, des pairs d'objets. Toutes les paires possèdent une propriété unique, mais du même type. Chaque objet de chaque pair est éparpillé dans la scène de jeu. Le but du joueur est de rassembler les pairs.
- **Gameplay contextualisé :** Le joueur doit attraper des petites créatures de couleurs unies pour les placer dans des fioles de couleurs correspondantes.
- **Durée:** Moyenne. 8~10 secondes
- **Narration :** Un scientifique expérimente sur des mini-créatures sentientes. Elles passent leur vie dans une boîte jusqu'au jour tant redouté où elles sont mises de force dans des fioles pour être testées.

Caméra

2D Vue de côté

Permet l'implémentation d'un système de physique simple et robuste contribuant à un meilleur sentiment de jeu.

Contrôle



Souris

L'entièreté du jeu est contrôlé uniquement avec la souris. Seul le mouvement et le clic gauche sont utilisé en tant qu'inputs.

Personnage

Main

Le joueur incarne la main du scientifique qui pourras attraper et placer les créatures dans les fioles.

✓ Condition de Victoire

Toutes les créatures ont été placées dans leurs fioles correspondantes à la fin du timer.

✗ Conditions de défaite

Une créature a été placée dans la mauvaise fiole.

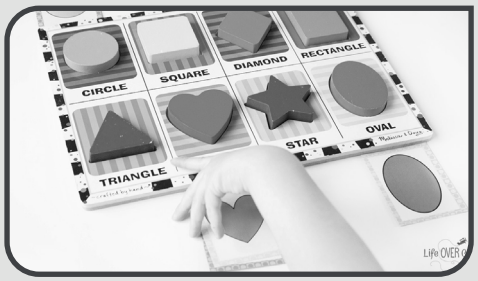
OU

Une ou plusieurs créatures n'ont pas étés placées à la fin du timer.

Références gameplay :

Machine à pinces.

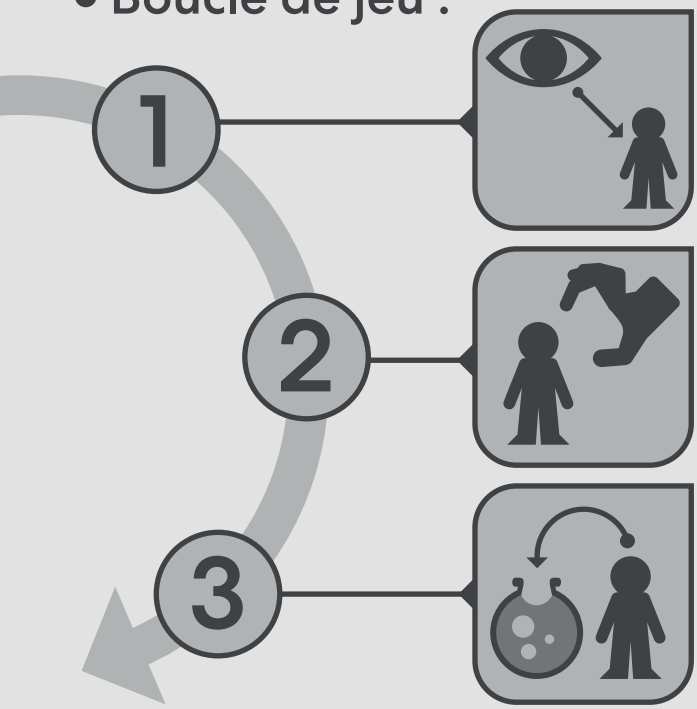
Il est important de répliquer la satifisfaction trouvée quand l'on arrive à attraper un jouet.



Jeu de correspondance des formes.

Le but du jeu cherche à être aussi simple que ces jeux pour enfants.

• Boucle de jeu :



Repérer une cible

Premièrement, le joueur doit repérer la créature qu'il compte attraper. Cette tâche n'implique que de ***l'observation*** et peut être compliqué par certaines actions faites par les créatures. Par exemple, une créature peut se cacher derrière certains objets pour être moins visibles.

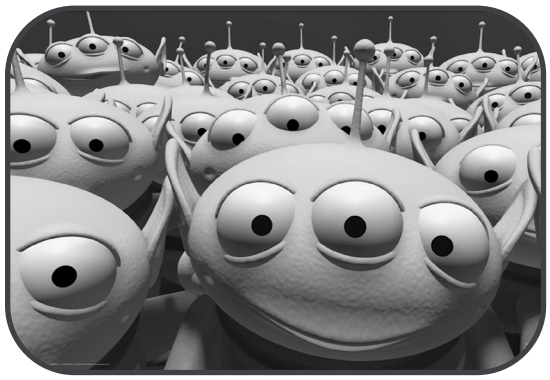
Attraper la cible repérée

Ensuite, le joueur doit ***déplacer sa souris*** de telle sorte à correctement positionner la main au-dessus de la créature. À partir de ce moment, le joueur peut ***effectuer un clic gauche*** pour saisir la créature. Tant que le clic gauche reste enfoncé, la créature reste attrapée.

Placer la cible capturée

Enfin, le joueur peut amener la créature près de la fiole de la couleur correspondante. ***En relâchant le clic gauche***, la créature tombe dans la fiole. Le joueur peut aussi tenter de lancer la créature dans la fiole pour essayer de gagner du temps.

Références d'ambiance :



Aliens : Toy Story

Au niveau des créatures, le principe est d'avoir un comportement au total inverse des aliens dans Toy Story. Dans ce cas, la main remplace le grapin. Cependant ici elle n'est pas un dieu bénévolant, mais au contraire une menace terrifiante.

Ils courent dans tous les sens pour essayer de ne pas se faire attraper et cri de peur s'ils le sont.

Tom & Jerry

Le côté physiqué du jeu permet de rajouter des éléments proche de la ***"slapstick comedy"*** qu'on peut trouver dans Tom & Jerry. Le joueur peut lancer une créature pour l'envoyer directement dans une fiole.

Les animations jouent un rôle très important pour correctement convier cet humour.

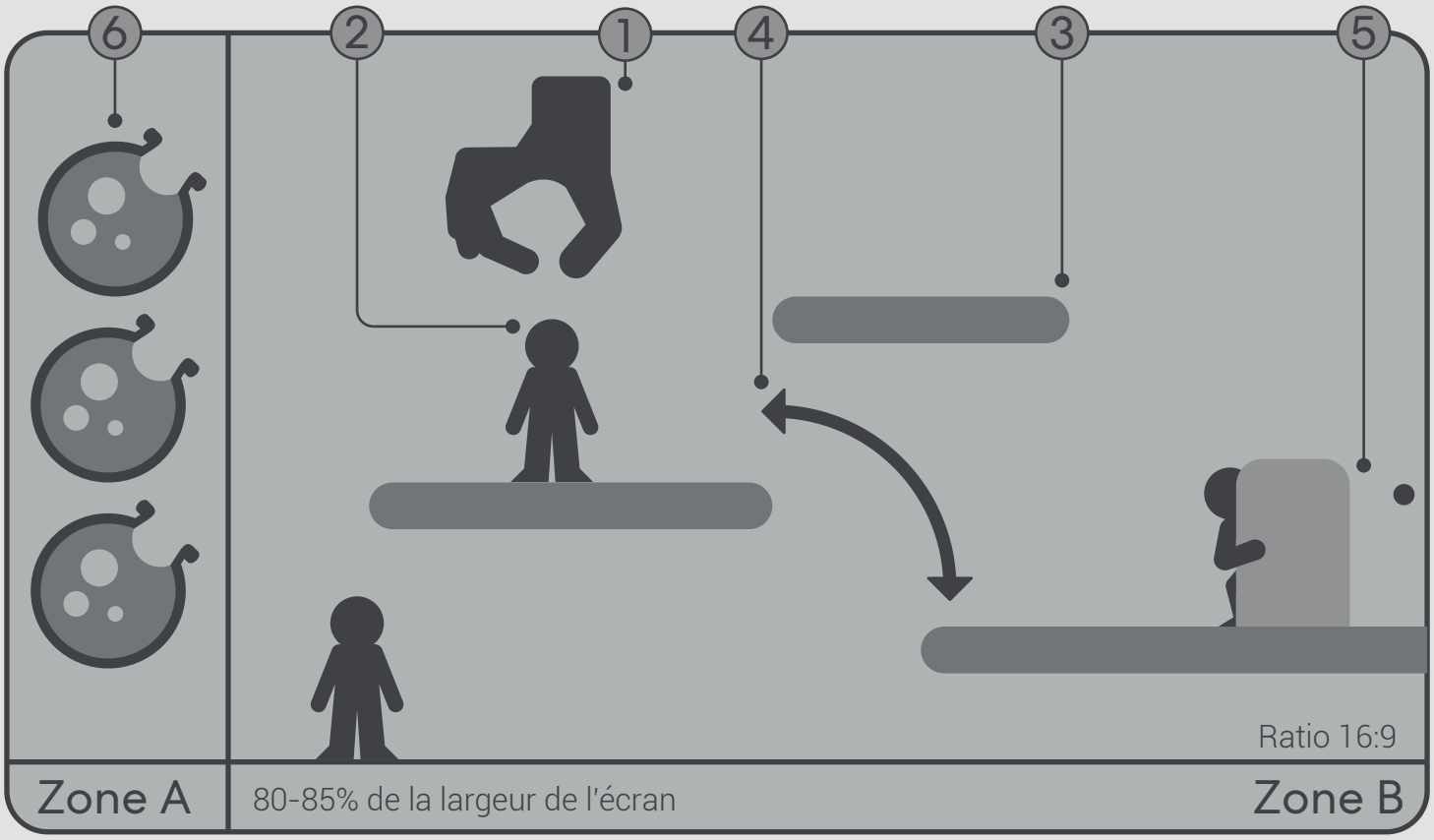
"* you ****face"*** **: Her - Spike Jonze**

Pour les séances d'intro et d'outro du micro-jeu, il sera possible de rajouter des animations supplémentaires.

Le but serait d'y placer un humour plus vulgaire où les créatures font des gestes obscènes envers le joueur comme pour le défier ou se moquer.

GAME OBJECTS

Agencement de l'Écran & GameObjects :



- **Zone A :** Le joueur ne peut entrer dans cette zone que de façon limitée pour placer les meeps dans les fioles. Ainsi, les informations présentes dans cette zone ne risquent pas d'être obstrué. Pour cette raison, les informations comme le temps restant seront affichés dans cette zone.
- **Zone B :** Cette zone constitue l'espace principale d'interaction. De plus les meeps ne peuvent se déplacer qu'a l'intérieur de cette zone.

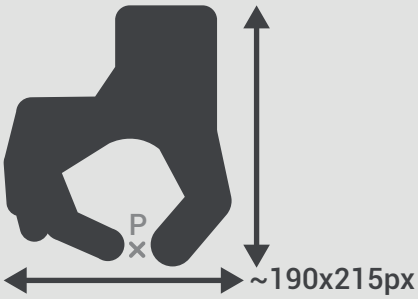
1. **Gamer hand :** Le personnage joueur. Sa position est contrôlée par la position de la souris. La gamer hand peut attraper et relâcher les meeps.
2. **Meep :** Les meeps sont les créatures que le joueur doit attraper et placer dans les fioles. Chaque meep possède une couleur unique. Leurs comportements évoluent en fonction du niveau de difficulté.
3. **Plateforme :** Zones de déplacement horizontal pour les meeps. Les plateformes ne sont pas un obstacle pour la gamer hand.
4. **Pont :** Points de traversée qu'un meep peut emprunter pour passer d'une plateforme à une autre.
5. **Cachette :** Élément apparaissant à partir du deuxième palier de difficulté et permettant au meep de se cacher partiellement.
6. **Fiole :** Contenant dans lequel les meeps peuvent être placé. Une fiole ne peut pas accueillir plus qu'un seul meep.

GAMER HAND

Résumé :

La gamer hand est le personnage joueur. Sa position est contrôlée par la souris. Elle permet au joueur d'attraper les meeps et de les placer dans les fioles.

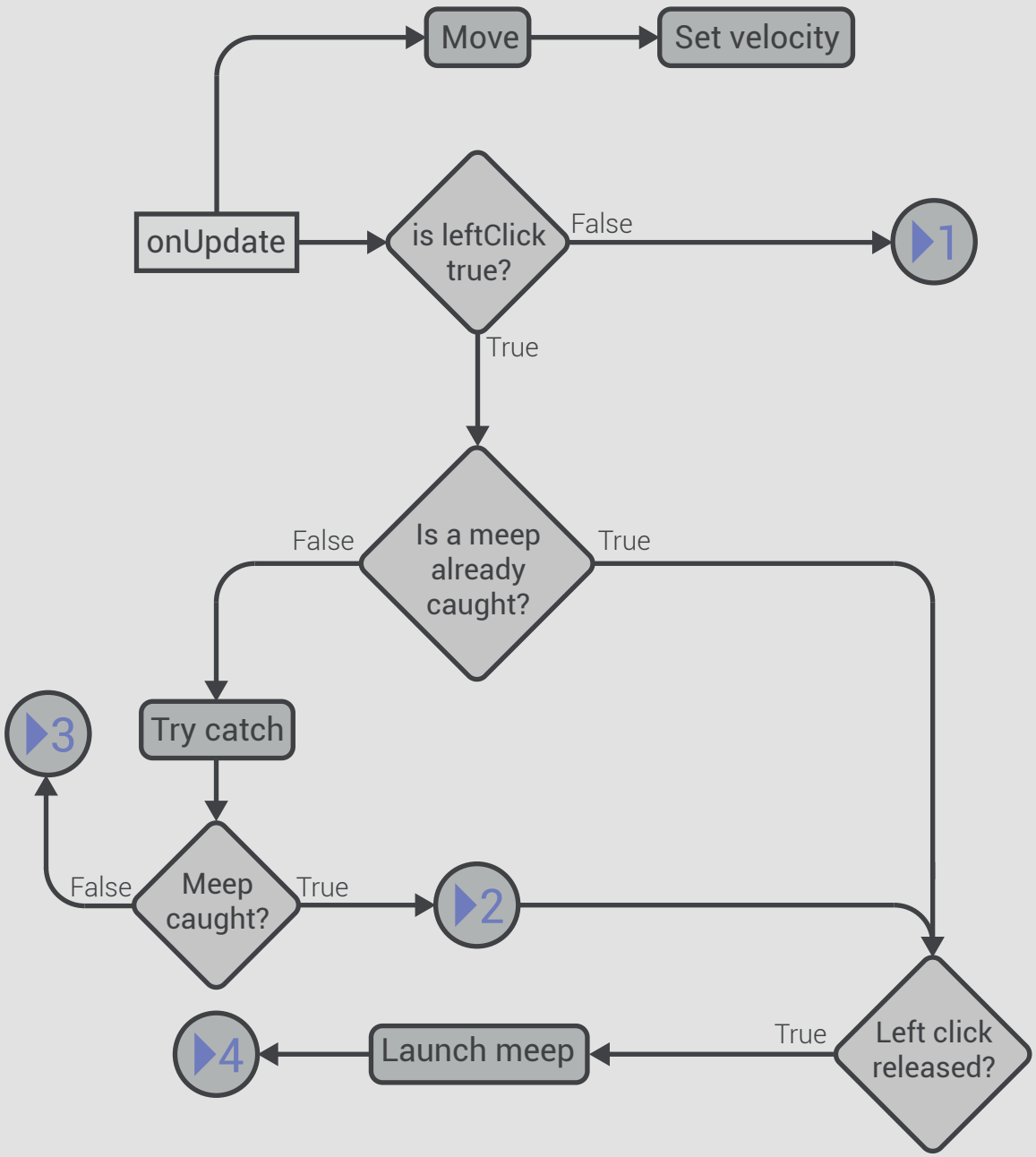
Format :



Inputs & Variables :

- *Vector3* mousePosition
- *Bool* leftClick
- *float* launchForce
- *float* speed
- *float* catchRange

Flowchart :



• 1 Bouger :

Move

-P : point de pivot du sprite de la gamer hand.
-mousePosition : position de la souris sur l'écran.
-speed : vitesse à laquelle la gamer hand se déplace pour atteinre la ***mousePosition***

placement de P

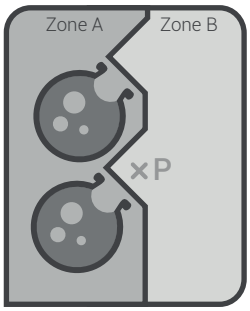


Schéma 1

La gamer hand ne peut être déplacé qu'au sein de la ***zone B***. La délimitation entre la *zone A* et *B* se fait selon le schéma 1.

Si la ***mousePosition*** sort de la ***zone B***, P prendra comme position le point le plus proche de la ***mousePosition*** dans la ***zone B***. Dans le cas du schéma 2, ce point est ***P+1.***

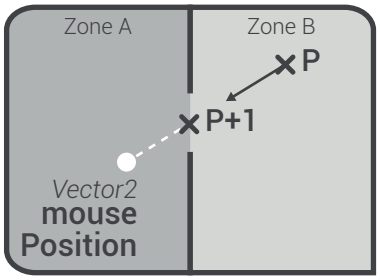
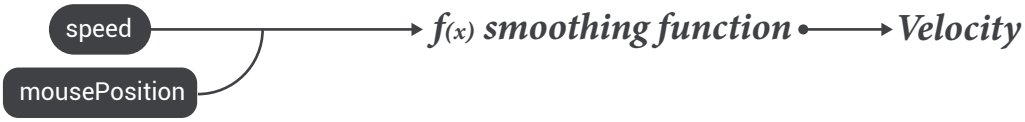


Schéma 2

La délimitation particulière de la zone B permet au joueur d'avoir un meilleur accès aux fioles pour placer les meeps.

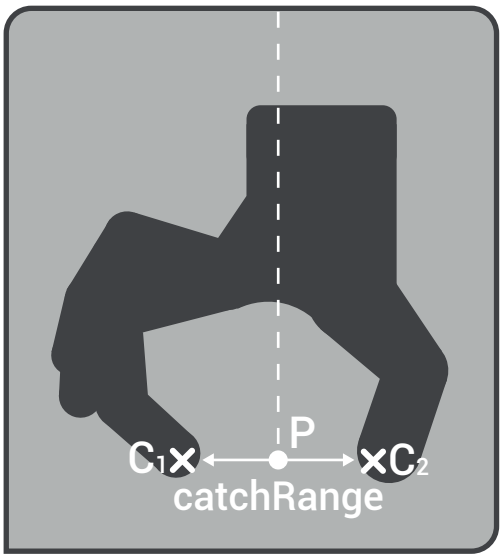


La position de la gamer hand ne prend pas directement la position de la souris. Le but est de s'en rapprocher graduellement en fonction de la variable de vitesse pour créer un mouvement lisse. Le mouvement de la gamer hand doit tout de même être assez snappy pour permettre au joueur d'être précis dans ses actions.

• 2 Attraper:

Try catch

-catchRange : écart entre l'index et le pouce de la gamer hand. Permet avec *P*, le calcul de l'espace de capture des meeps.



Espace de capture

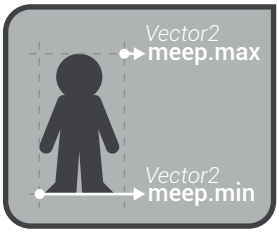
$C_1 = P - (catchRange / 2)$
$C_2 = P + (catchRange / 2)$

Dès que le clic gauche est enfoncé, un meep peut être immédiatement attrapé si les conditions suivantes sont remplies :

- Le meep est à hauteur de *P*.
if (P.y > meep.min.y && ***P.y < meep.max.y)***

-Le meep se trouve entre C_1 et C_2.
if (meep.x > C1.x && ***meep.x < C2.x)***

Meep bounds



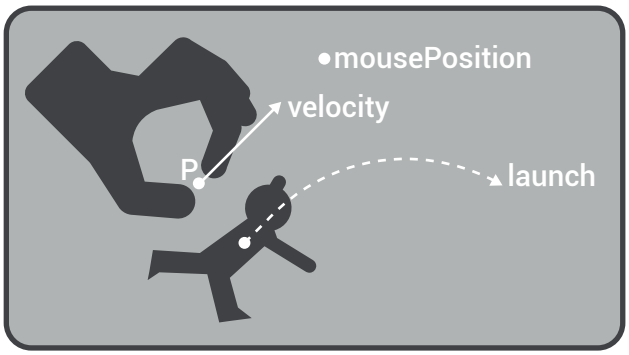
Consignes : ⚠

-Si une capture de meep est ratée, le joueur peut directement cliquer à nouveau pour réessayer.
-Tant que le clic gauche est enfoncé, un meep reste attrapé et ne peut pas être libéré.
-Avoir attrapé un meep n'impacte pas la variable de vitesse.
-La gamer hand ne peut pas attraper plusieurs meeps à la fois.

• 3. Relâcher :

Launch meep

-launchForce : force avec laquelle un meep est propulsé quand le joueur le relâche le clic gauche.



Si le joueur a attrapé un meep et qu'il relâche le clic gauche, le meep sera lancé dans la direction du mouvement de la gamer hand selon le calcul ci-dessous.

$$launch = velocity * launchFore$$

Relations gameplay :
Cette mécanique offre deux approches au joueur pour placer un meep dans une fiole :

- Placer la gamer hand au-dessus d'une fiole pour y laisser doucement tomber le meep car ce dernier est éjecté avec une vélocité quasi inexistante.
- Effectuer un mouvement de souris large et rapide durant lequel le clic gauche est relâché, ce qui as pour effet de lancer le meep qui, si l'action est bien effectué, tomberas dans la bonne fiole

Références graphiques :

MasterHand : Nitendo

Style pixel art + résolution

Protection pour manipulation de virus

Animations & Sons :

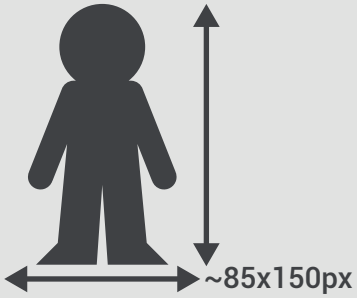
	Description	Animation	Son
1 Idle	la gamer hand bouge doucement de haut en bas	Optionnel	Optionnel
2 Sucessful catch	la gamer hand tiens, entre l'index et le pouce un meep.	Oui	Oui
3 Failed catch	la gamer hand tente d'attraper un meep mais rate, l'index et le pouce se frappes l'un contre l'autre.	Oui	Oui
4 Meep launch	la gamer hand fait un geste de lancer avec le poignet et lâche le meep qu'elle tiens.	Oui	Optionnel

MEEP

Résumé :

Les meeps sont les créatures que le joueur doit attraper et placer dans les fioles. Chaque meep possède une couleur unique. Leurs comportements évoluent en fonction du niveau de difficulté.

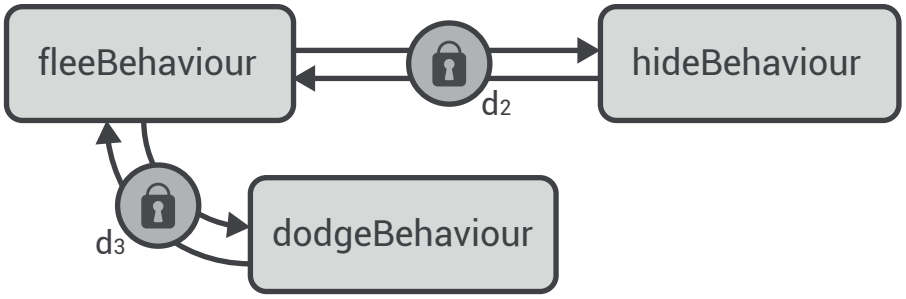
Format :



Variables :

- *float* speed
- *float* dodgeRadius
- *float* dodgeChance
- *float* turnChance — Range 0 - 1
- *float* hideTime
- *int* showCount
- *Vector2* turnPosRange
- *Color* color

Finite State Machine :



Le schéma ci-dessus indique les différents comportements qu'un meep peut adopter.

-fleeBehaviour : Le meep cours sur les plateformes. ⟶ débloqué au palier *(d1)*.

-hideBehaviour : Le meep essaye d'atteindre une cachette puis s'y cache. ⟶ débloqué au palier *(d2)*.

-dodgeBehaviour : Le meep évite le mouvement de la gamer hand. ⟶ débloqué au palier *(d3)*.

Couleurs d'instantiations :

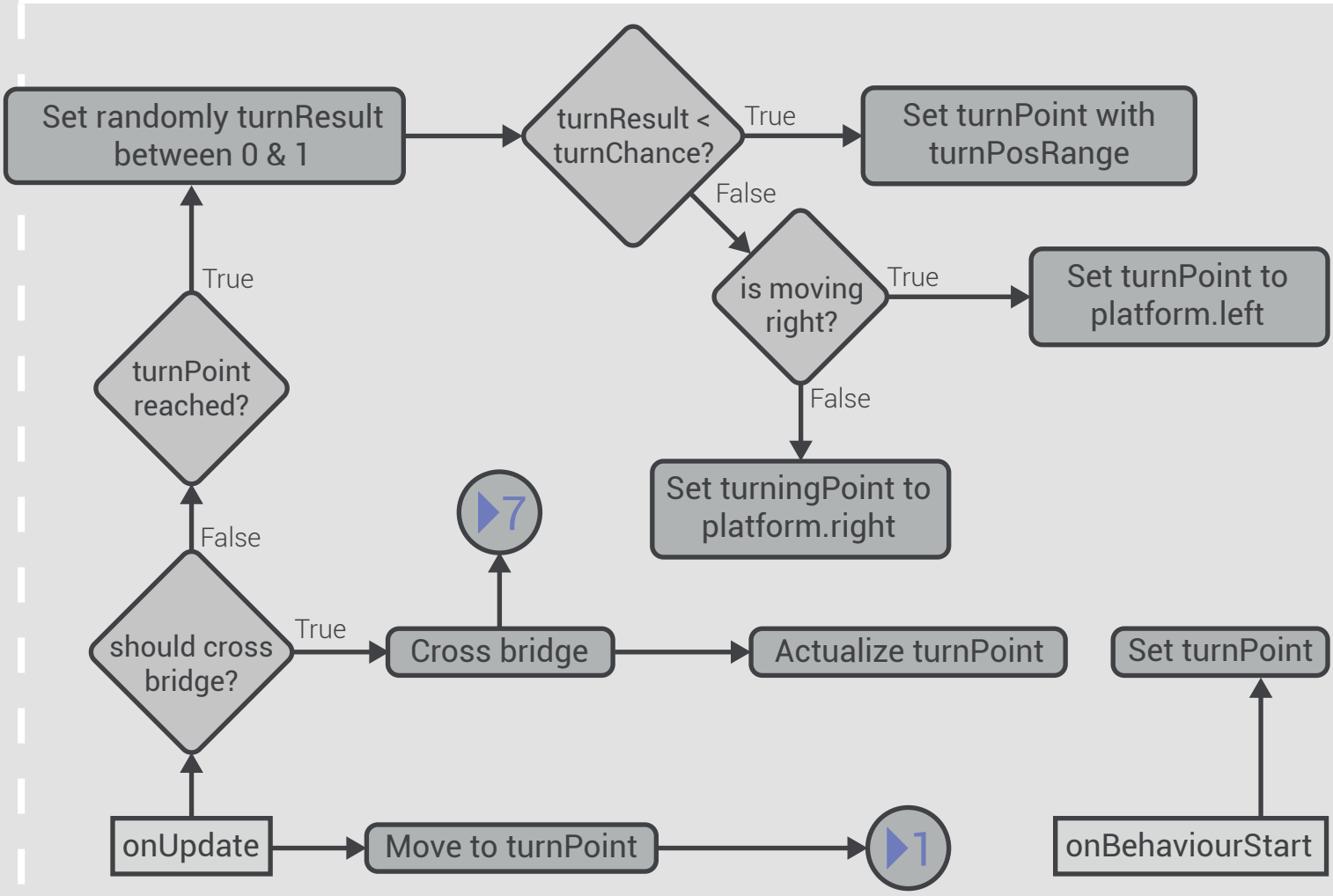
Au début du micro-jeu. Chaque meep se voit assigner une couleur unique parmis la liste ci-dessous. Plusieurs meeps ne peuvent pas partager la même couleur.

meepColors Palette

	Hex	RGB
	#069292	*r:* 006 / *g:*146 / *b:* 146
	#8D71B0	*r:* 141 / *g:*113 / *b:* 176
	#DA6C11	*r:* 218 / *g:* 108/ *b:* 017
	#6AB42D	*r:* 106 / *g:* 180 / *b:* 045
	#79B1E0	*r:* 121 / *g:* 177 / *b:* 224
	#ED71A8	*r:* 237 / *g:* 113 / *b:* 168

• 1. FleeBehaviour :

La ***fleeBehaviour*** est le comportement de base d'un meep qui lui permet de bouger au sein du niveau est de ne pas être trop facile à attraper.



- **speed :** la vitesse à laquelle le meep se déplace
- **turnPoint :** *(Vector2)* la position que le meep doit atteindre avant d'inverser son sens de mouvement.
- **turnChance :** la chance q'un meep a d'inverser son sens de mouvement avant d'atteindre l'extrémité d'une plateforme.
- **turnResut :** *(float)* valeur générée aléatoirement entre 0 et 1.
- **turnPosRange :** une valeur x et y définissant un minimum et un maximum tout deux contenu entre 0 et 1. Cet intervalle permet de déterminer où est ce qu'un meep change de sens sur une plateforme.

Durant la ***fleeBehaviour***, un meep a pour but constant de se rendre jusqu'à son turnPoint actuel. Une fois cette objectif atteint, ***turnResult*** est comparée à ***turnChance*** dans le but d'actualiser ***turnPoint***.

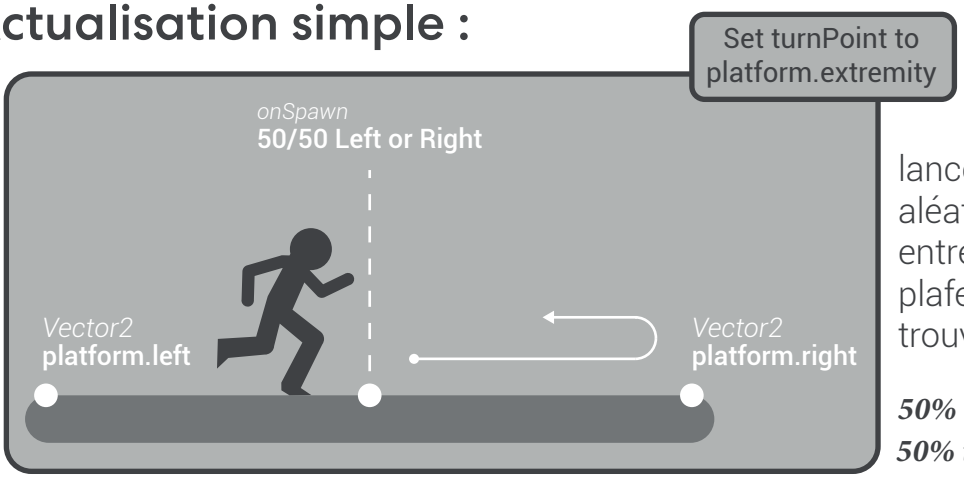
Move to turnPoint

Set randomly turnResult between 0 & 1

- Si ***turnResult*** est inférieur à ***turnChance*** : **Actualisation simple.**

- Si ***turnResult*** est supérieur à ***turnChance*** : **Actualisation aléatoire.**

Actualisation simple :



Set turnPoint

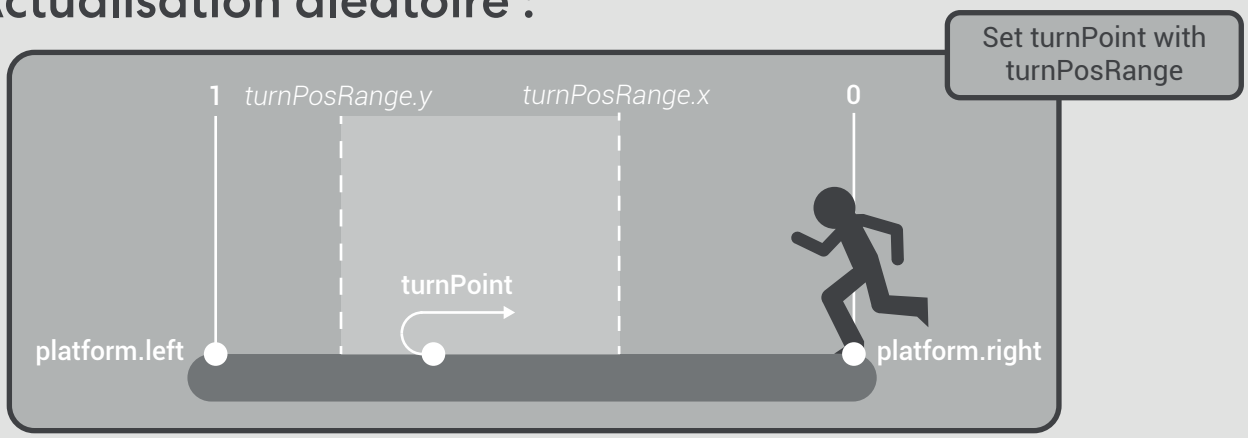
Quand la ***fleeBehaviour*** est lancé, le ***turnPoint*** est déterminé aléatoirement à chance égale entre les deux extrémités de la plafeforme sur laquelle il se trouve.

50% turnPoint = platform.right
50% turnPoint = platform.left

Pour l'actualisation simple, le ***turnPoint*** devient l'extrémité droite de la plateforme si le meep va à gauche ou l'extrémité gauche s'il va à droite.

Par exemple, pour le schéma ci-dessus le meep cours vers la droite et son ***turnPoint*** actuel est ***platform.right***. Quand il aura atteint ce point, le prochain ***turnPoint*** pour une actualisation simple serait ***platform.left***.

Actualisation aléatoire :

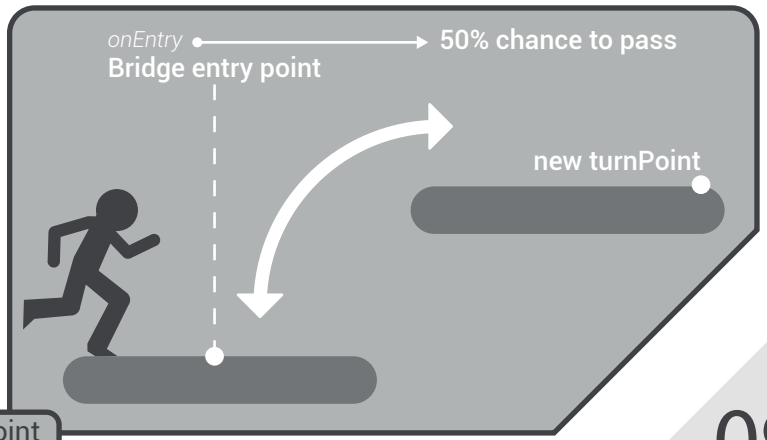


Pour l'actualisation aléatoire, on prend en premier le nouveau point de ***turnPoint*** pour une actualisation simple. Ensuite, une étendue est créée entre ce point et la position du meep et on génère un pourcentage aléatoire entre ***turnPosRange.x*** et ***turnPosRange.y*** Ainsi ***turnPoint*** devient le point égale à l'étendue multiplié par le pourcentage.

Dans le cas du schéma ci-dessus, le calcul serait :
turnPoint = platform.right + ((platform.left - platform.right) * rand(turnPosRange.x, turnPosRange.y))

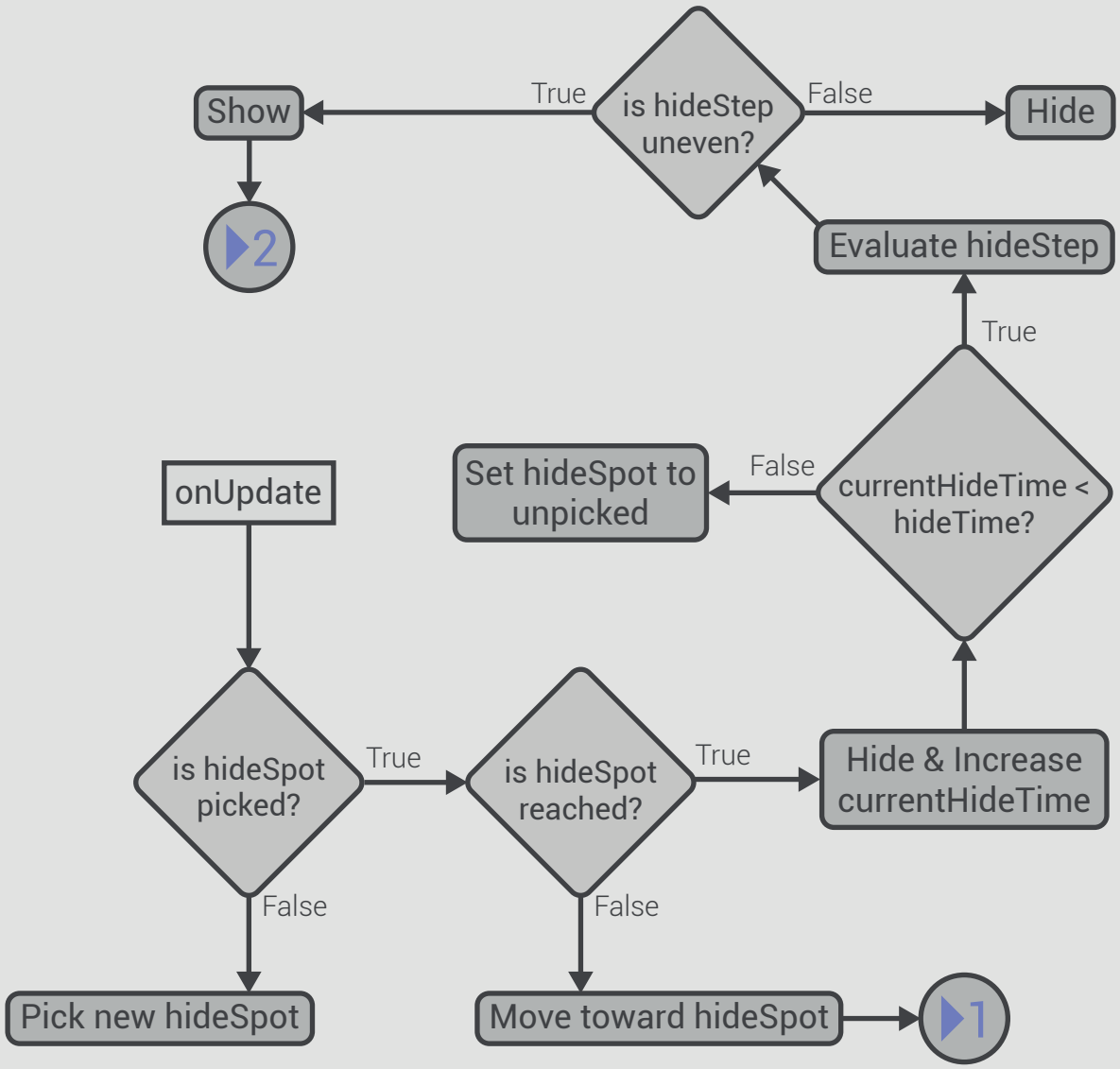
Emprunt de ponts :

Dans le cas où un meep rencontre un pont sur son chemin. Il a 50% chance de l'emprunter. Si l'emprunt se fait, le ***turnPoint*** est actualisé et devient l'extrémité droite de la plateforme d'arrivée si le meep va à droite ou l'extrémité gauche s'il va à gauche



• 2. HideBehaviour :

La ***hideBehaviour*** est le comportement permettant à un meep de se dissimuler dans une cachette pour être moins facilement repéré par le joueur. Ce comportement est débloqué pour tous les meeps du micro-jeu au deuxième palier de difficulté.



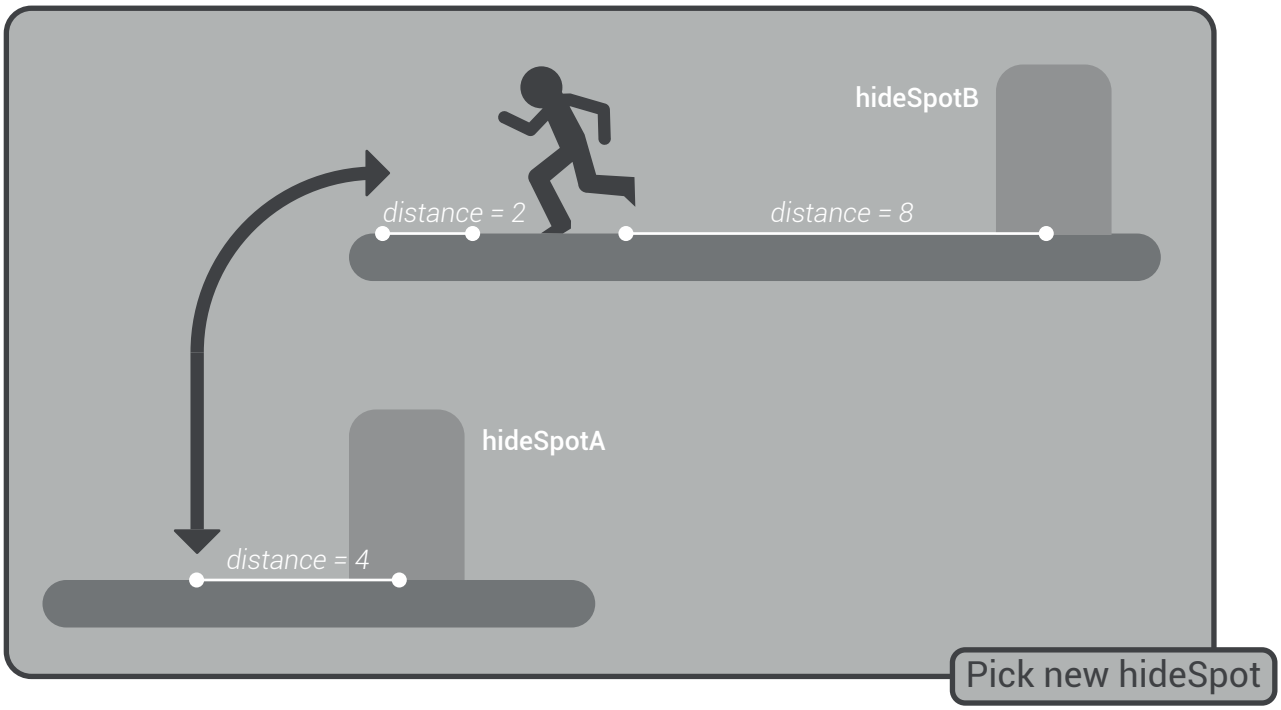
- **hideTime :** le temps q'un meep reste derrière une cachette.
- **currentHideTime :** *(float)* le temps q'un meep a passé à être caché.
- **showCount :** le nombre de fois q'un meep révèle sa présence quand il est entrain de se cacher.
- **hideStep :** *(int)* valeur permettant de déterminer si le meep doit se révéler ou non.
- **hideSpot :** *(Vector2)* position de la cachette où un meep ira se cacher.

Move toward hideSpot

Durant la ***hideBehaviour***, un meep cherche à atteindre le ***hideSpot*** qui lui a été assigné pour s'y dissimuler.

Une fois son ***hideSpot*** atteint, un meep y restera pour le temps indiqué par ***hideTime*** et révélera une partie de son sprite un nombre de fois égale à ***showCount***. Une fois le temps de cachette atteint, un meep se verra assigner un nouveau ***hidingSpot***.

Choix de la cachette :

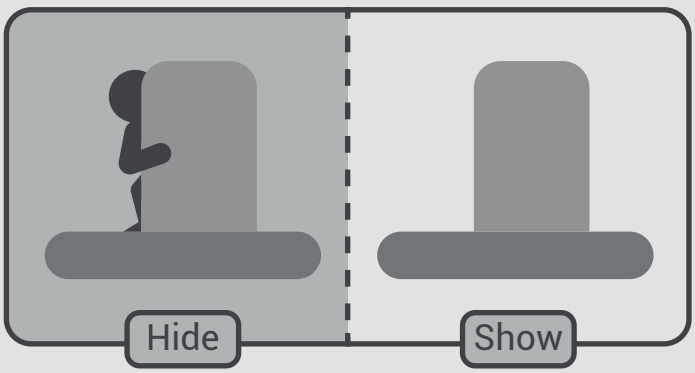


Au début de la ***hideBehaviour*** d'un meep ou quand son ***currentHideTime*** devient égale ou supérieur à ***hideTime***, il est nécessaire de lui assigner un nouveau ***hideSpot***. Le nouveau ***hideSpot*** d'un Meep est toujours celui inocupé et le plus proche de lui. Les ponts ne rentrent pas en compte de le calcul de distance.

Par exemple, pour le schéma-ci dessus, le ***hideSpot*** choisi pour le meep serait ***hideSpotA,*** car il est à une distance de 6 comparé à ***hideSpotB*** se trouvant à une distance de 8 du meep.

Indice visuelle sur les meeps cachés :

Evaluate hideStep ***hideStep = ceilToInt((currentHidingTime / hidingTime) * (showCount * 2))***



Quand un meep est entrain de se cacher, son ***currentHideTime*** est incrémenté chaque frame par le temps en seconde s'étant écoulé depuis la dernière frame.

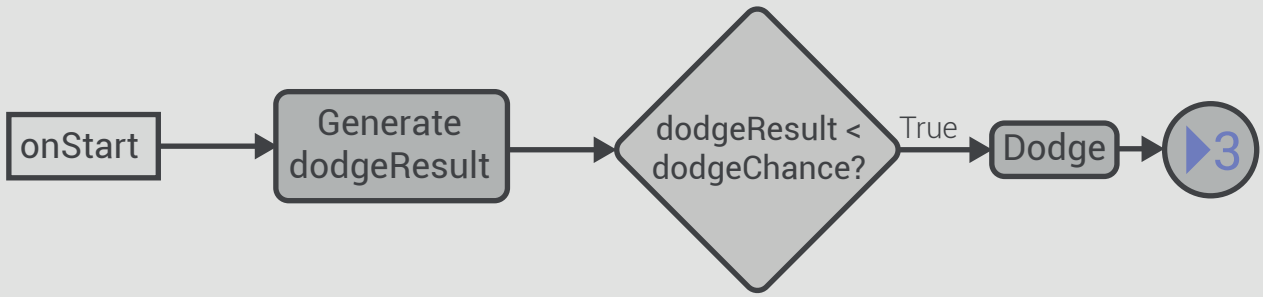
De plus ***hideStep*** est calculé selon le calcul ci-dessus. Si ***hideStep*** est pair, le meep se cache totalement *(Hide)*, sinon il révèle une partie de son sprite *(Show)*.

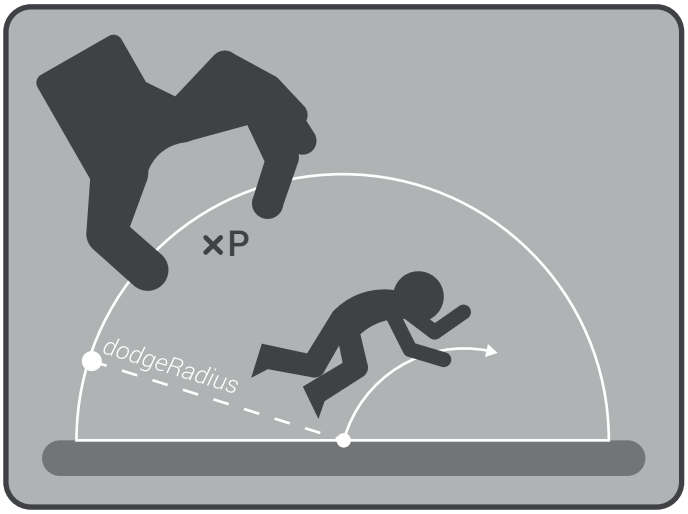
Hide & Increase currentHidingTime

• 3. DodgeBehaviour :

La ***dodgeBehaviour*** est le comportement permettant à un meep d'essayer d'éviter de se faire capturer en effectuant une roulade vers l'avant. Ce comportement est débloqué pour tous les meeps du micro-jeu au troisième palier de difficulté.

- **dodgeRadius :** rayon, autour du meep, dans lequel le point de pivot du sprite de la gamer hand doit se trouver pour que le meep puisses tenter une esquive.
- **dodgeChance :** chance qu'un meep a de faire une esquive.
- **dodgeResut :** *(float)* valeur générée aléatoirement entre 0 et 1.





Au début de la ***dodgeBehaviour***, **dodgeResult** est généré :

- Si ***dodgeResult*** est inférieur à ***dodgeChance***, le meep n'esquive pas et la ***dodgeBehaviour*** se finit.

- Si ***dodgeResult*** est supérieur à ***dodgeChance***, le meep effectue une esquive vers l'avant puis la ***dodgeBehaviour*** se finit.

• Tableau de transitions :

Le tableau ci-dessous indique quand est ce qu'une transition doit s'effectuer d'une behaviour à l'autre et le pourcentage de réussite nécessaire pour que cette transition s'effectue.

Par exemple, si un meep est dans sa ***fleeBehaviour*** et qu'il vient d'atteindre son ***turnPoint***, il a 50% de chance de passer à sa ***hideBehaviour***.

	Quand	Chance
Flee > Hide	le meep a atteint son turnPoint	50%
Hide > Flee	le meep a fini de se cacher	50%
Flee > Dodge	la gamerHand est rentré est à distance de dodgeRadius du meep et est au dessus du meep	100%
Dodge > Flee	le meep a finit sa roulade	100%

Précisions supplémentaires :

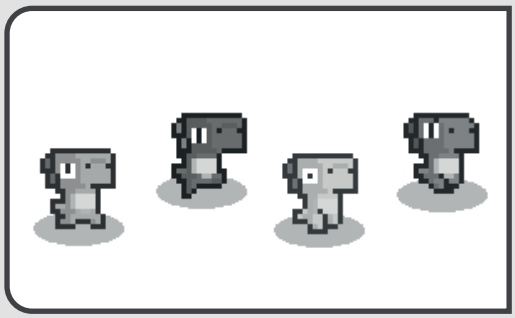
- Si un meep est attrapé par la gamer hand ou s'il est en train de tomber parcequ'il a été jeté par la gamer hand, aucune behaviour ne peut s'effectuer.
- Si un meep est lancé par la gamer hand et qu'il n'atterrit pas dans une fiole, il doit à tous les coups retomber sur une plateforme et commencer sa ***fleeBehaviour***.
- Un meep peut être attrapé même si il est caché.

Références graphiques :

*"**** you ****face"* : Her - Spike Jonze

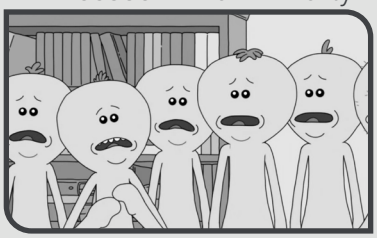
Personnage simple & unicolore :
Forme d'un meep

Dino Characters : Arks



~14 frames
Style pixel art & Animation complexity

Mr meessek : Rick & Morty



Mental breakdown :
Émotions transmises par l'animation

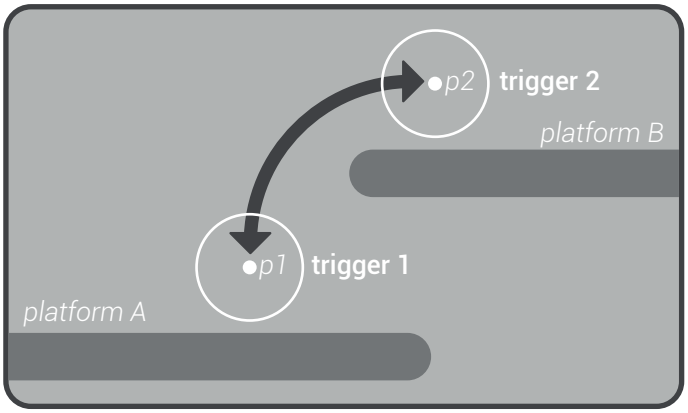
Human Fall Flat
Morphologie d'un meep

Animation & Sons :

	Description	Animation	Son
1 Run	le meep cours en panique vers l'avant avec les bras en l'air.	Oui	Non
2 Show	le meep est derrière une cachette et révéle sur le côté droit ou gauche une partie de son corps.	Oui	Non
3 Dodge	le meep s'élance en avant tout en faisant une roulade pour esquiver la gamerHand.	Oui	Non
4 Catched	le meep est attrapé entre l'index et le pouce de la gamer hand, il se débat frénétiquement.	Oui	Optionnel
5 Airborne	le meep est entrain de tomber en chute libre et a peur pour sa vie.	Oui	Oui
6 Mock	le meep fait n'importe quel geste obscène à l'intention du joueur.	Optionnel	Optionnel
7 Jump	le meep saute d'une plateforme à l'autre	Oui	Non

PONT

Vector2 p1 | Vector2 p2



Un pont est un set de deux trigger lié tout deux à une plateforme. Quand un meep rentre en collision avec l'un des deux trigger, il emprunte le pont. Ceci l'amène à bouger depuis la plateforme sur laquelle il est jusqu'à l'autre plateforme lié au pont.

p1 et ***p2*** sont les points où se placent respectivement *trigger1* et *trigger2*.

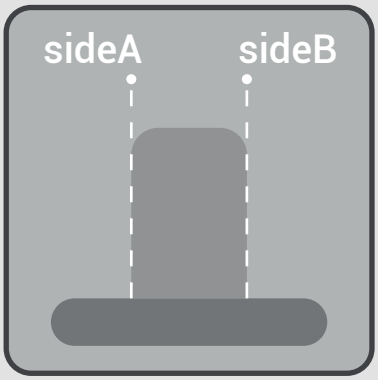
⚠ Il n'y a pas de représentation graphique pour les ponts, quand un meep emprunte un pont, le joueur le voit sauter de la 1ère plateforme à la seconde.

CACHETTE

Vector2 sideA | Vector2 sideB

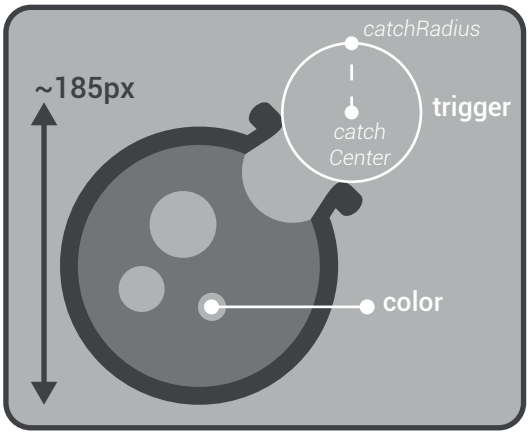
Une cachette est un objet de décor se détachant du background. Il peut être utilisé par les meeps pour se cacher. Tout sprite de cachette doit être rectangulaire pour déterminer ***sideA*** et ***sideB***.

Quand un meep effectue son animation *(Hide)* : S'il se revèle à gauche, son centre devient ***sideA*** et si il se révèle à gauche, son centre devient ***sideB***.



FIOLE

Color color | float catchRadius | Vector2 catchCenter



Contenant dans lequel les meeps peuvent être placé. Une fiole ne peut pas accueillir plus qu'un seul meep.

Il y a autant de fioles que de meeps. De plus chaque meep est lié à une fiole pour former des paires de couleurs. Il suffit qu'un meep entre en collision avec le *trigger* d'une fiole pour qu'il y soit placé. Si un meep est placé dans une fiole et que les deux n'ont pas la même couleur, le micro-jeu est perdu.

-***catchCenter*** est le point où se place le trigger d'une fiole.
-***catchRadius*** est le rayon du trigger d'une fiole.

DIFFICULTÉ

Principe :

Trois éléments clés jouent dans l'augmentation de la difficulté : l'augmentation du BPM, le nombre de meeps à attraper et la compléxité de leurs comportements.

• 1 Agumentation du BPM :

L'impact du BPM est perçu par rapport à la vitesse grimpante des meeps. La gamer hand ne voit pas sa vitesse augmentée en fonction du BPM. Ceci fait que le joueur auras plus de mal a attrapé les meeps si le BPM est élevé.

$$meep.speed = BPM * ratio$$

• 2 Nombre de meeps :

Le micro-jeu à sa difficulté la plus basse, a 2 meeps que le joueur doit attraper. Pour chaque augmentation du palier de difficulté, un meep est rajouté.

	Nombre de Meeps
d1 : **Difficulté basse**	2
d2 : **Difficulté moyenne**	3
d3 : **Difficulté élevé**	4

• 3 Comportement évolutif des meeps :

Pour chaque palier de difficulté, les meeps débloque un nouveau comportement qui les rend plus compliqués à attraper.

	Behaviours disponibles
d1 : **Difficulté basse**	flee
d2 : **Difficulté moyenne**	flee / hide
d3 : **Difficulté élevé**	flee / hide / dodge

Ajustement par réglage de variables :

Les 3 principes de difficulté mentionnés ci-dessus sont à respecter et ne doivent pas être altérés pour rendre la complétion du micro jeu plus simple. Dans le cas où le micro-jeu semble trop dure, les variables suivantes peuvent être modifié pour le rendre plus simple.

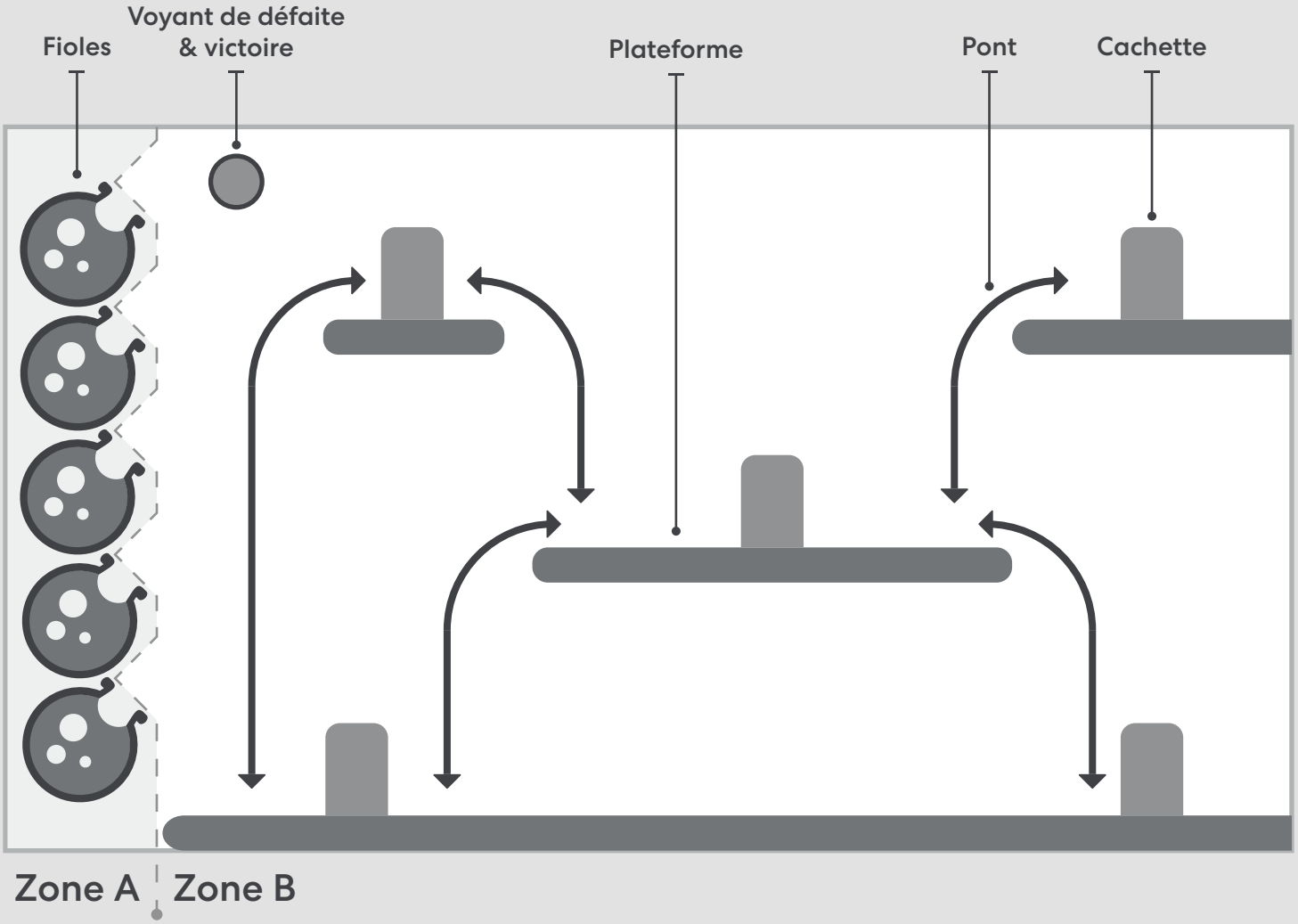
	Modification
gamerHand.catchRange	augmentation
gamerHand.speed	augmentation
meep.dodgeChance	réduction
meep.hideTime	réduction
meep.showCount	augmentation
fiole.catchRadius	augmentation

LEVEL DESIGN

Le level design présenté ci-dessous est le seul disponible pour ce micro-jeu. L'emplacement des ponts et des plateformes ne changent en aucun cas. Pour ce qui est des cachettes, elles n'aparaissent que si le micro-jeu est réglé au deuxième niveau de difficulté.

Pour ce qui est des fioles, leur nombre est égal au nombre de meeps dans le niveau. La première fiole en partant du haut est toujours associé au meep blanc qui se fait attraper et placer durant la séquence d'intro du micro-jeu.

Ce level design peut être ajusté si besoin. Il est possible que l'emplacement des cachettes ou des ponts ne soit pas optimale. Ces éléments doivent être changés si un playtest en confirme la nécessité.



INTRO & OUTRO

Introduction :

On voit la scène de jeu avec les meeps vaquant paisiblement à leurs occupations. Soudain, la gamer hand s'immisce dans l'écran et tous les meeps commencent à paniquer. La gamer hand en attrape un et le place dans sa fiole correspondante.

Le verbe d'action ***"Sort'Em !"*** apparait en grand sur l'écran. À la disparition du verbe d'action, le joueur prend le contrôle de la gamer hand et les meeps commencent à courir dans tous les sens.

Scène de fin de victoire :

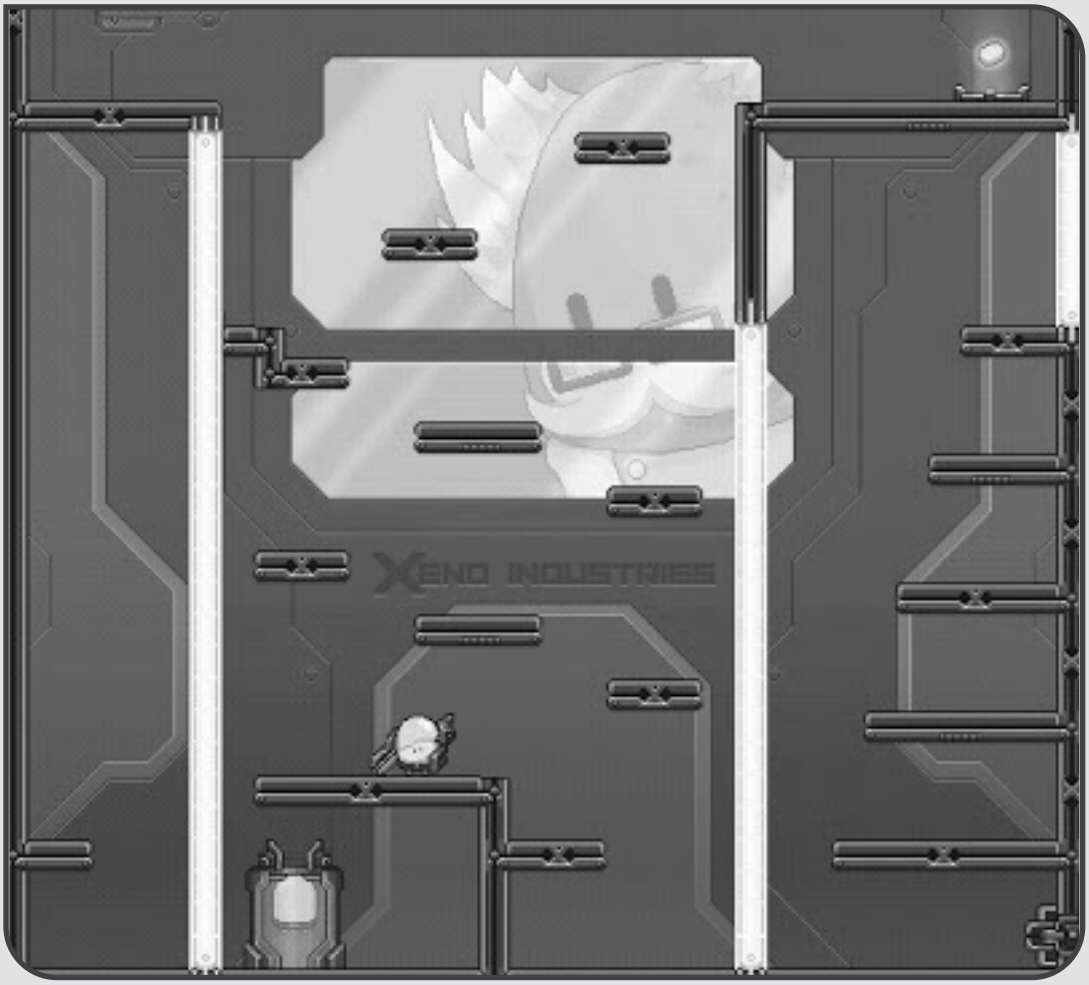
Le joueur perd contrôle de la gamer hand et elle sort de l'écran. Le support coulissant contenant les fioles se rétracte et le voyant lumineux en haut à gauche de l'écran s'allume en vert.

Scène de fin de défaite :

Le joueur perd contrôle de la gamer hand et elle sort de l'écran. Le support coulissant contenant les fioles se rétracte et le voyant lumineux en haut à gauche de l'écran s'allume en rouge. Durant toute cette séquence, les meeps se moquent du joueur.

DIRECTION ARTISTIQUE

Intérieur de la meepBox



Fiole

Tirroir coullisant :
(contient les fioles)

Style pixel art + résolution

Labo dans lequel trouve la meepBox

Voyant de défaite / victoire

Meep

Gamer Hand

Objets :

Ci-dessous, la liste des différents objets et de leurs variables principales :

Gamer Hand

	Type	Utilité
speed	float	vitesse à laquelle la gamer hand se rapproche de la position de la souris
catchRange	float	espace entre l'index et le pouce de la gamer hand dans lequel un meep doit se trouvé pour être capturé
launchForce	float	force avec laquelle un meep est projeté dans les airs quand la gamer hand le lâche.

Meep

	Type	Utilité
speed	float	vitesse à laquelle le meep se déplace
color	color	couleur unique du meep choisi parmis la palette meepColors.

FleeBehaviour

	Type	Utilité
turnChance	float	chance qu'un meep a de changer de sens avant d'atteindre l'extrémité d'une plateforme
turnPosRange	float	intervalle définissant où peut se trouver, sur une plateforme, le point qu'un meep doit atteindre avant de changer de sens

HideBehaviour

	Type	Utilité
hideTime	float	temps qu'un meep reste dissimulé dérrière une cachette.
showCount	float	nombre de fois qu'un meep révèle une partie de son sprite pendant qu'il est caché

DodgeBehaviour

	Type	Utilité
dodgeChance	float	chance qu'un meep a d'éviter la tentative de capture de la gamer hand.
dodgeRadius	float	distance à laquelle la gamerHand doit se trouver, en plus d'être au dessus d'un meep pour que ce dernier puisses tenter une esquive.

Plateforme

	Type	Utilité
left	Vector2	extrémité gauche de la plateforme
right	Vector2	extrémité droite de la plateforme

Pont

	Type	Utilité
p1	Vector2	position du premier trigger d'un pont
p2	Vector2	position du second trigger d'un pont

Cachette

	Type	Utilité
sideA	Vector2	position qu'un meep prend si il révèle une partie de son sprite du côté gauche d'une cachette
sideB	Vector2	position qu'un meep prend si il révèle une partie de son sprite du côté droit d'une cachette

Fiole

	Type	Utilité
catchCenter	Vector2	position du trigger d'une fiole
catchRadius	Vector2	rayon du trigger d'une fiole
color	Color	couleur de la fiole associé à un seul meep dans le niveau

Initialisation du micro-jeu :

Ci-dessous, les étapes à suivre au lancement du micro-jeu.

1. Faire apparaître autant de meep que le niveau de difficulté indique à des endroits aléatoires dans le niveau.

2. Assigner à chaque meep une couleur unique provenant de la palette meepColors

3. Ajouter un meep de couleur blanche qui sera celui attrapé par la gamer hand dans la séquence d'intro.

4. Faire apparaître autant de fioles qu'il y a de meeps et assigner à chaque fiole la couleur d'un unique meep.

5. Lancer la séquence d'intro.

LEXIQUE

GameObject :

Objet de jeu ayant une position, rotation et taille dans l'espace et auquel des systèmes peuvent être attachés pour lui donner un but précis.

Sprite :

Graphique 2D possédant un point de pivot et un centre.

Finite State Machine (FSM) :

Assemblement logique de comportements où un seul comportement peut être actif à la fois. De plus, le comportement actif définit les transitions possibles vers d'autres comportements.

Behaviour :

Comportement d'une Finite State Machine.

Frame :

Temps s'étant écoulé depuis le dernier rafraîchissement de l'écran.

Trigger :

Forme 2D définissant un espace pouvant détecter les collisions, mais n'y participant au niveau physique.

Flowchart :

Diagramme définissant une suite d'actions logique.

MeepBox :

Boite dans laquelle se trouve les meeps.

FLOWCHART GUIDE

- **Action**

- **Callback**

Point d'entrée de la flowchart. Démarre la suite d'actions liées au moment spécifié par le type du callback :

-onStart : Au démarrage du micro-jeu.
-onBehaviourStart : Au démarrage d'un comportement de FSM.
-onUpdate : Chaque frame.

- **Condition**

Condition définissant l'embranchement de la suite d'actions. Si la condition est vérifiée, la suite d'actions prend la branche true, sinon la branche false est empruntée.

- **Lecture d'un clip**

Spécifie que la paire animation & son numéroté X et appartenant au gameObject lié au flowchart doit être joué.

ANNEXES