Le roi des neiges

Projet ceinture jaune

- instructions conditionnelles
- boucles
- capteurs
- opérateurs

But du jeu : Le joueur dirige *Snowman* en bas de la scène. *Snowman* doit éviter *Sun* qui le fait fondre, et essayer, au contraire, d'être touché par *Snowflake* qui le fait grossir. Le joueur gagne si *Snowman* devient assez gros et perd s'il devient trop petit.

Étape 1 *Snowman*

Le lutin *Snowman* doit se déplacer horizontalement en bas de la scène selon le mouvement de la souris.

- Comment faire pour que le lutin ne se déplace qu'horizontalement ?

.....

.....

.....

.....

.....

Étape 2 *Snowflake* et *Sun*

Les lutins *Snowflake* et *Sun* doivent se déplacer verticalement du haut de la scène vers le bas. Lorsqu'ils touchent *Snowman* ou atteignent le bas de la scène, *Snowflake* et *Sun* doivent réapparaitre dans une position aléatoire en haut de la scène et se déplacer à nouveau verticalement vers le bas.

- Quel problème rencontre-t-on si on utilise les commandes 

pour que le lutin *Sun* se déplace verticalement vers le bas ?

.....

- Rencontre-t-on le même problème avec le lutin *Snowflake* ? Pourquoi ?

.....

Étape 3 Interactions entre lutins

Les trois lutins doivent avoir approximativement la même taille au début du jeu. Lorsque *Snowman* est touché par *Sun*, sa taille doit diminuer, et *Sun* doit se placer en haut de la scène. Lorsqu'il est touché par *Snowflake*, sa taille doit augmenter, et *Snowflake* doit se placer en haut de la scène.

- Quel lutin doit-on réduire le plus par rapport à sa taille initiale ?

.....

- Quelles variations de la taille de *Snowman* peut-on choisir pour que le jeu soit intéressant à jouer ?

.....

.....

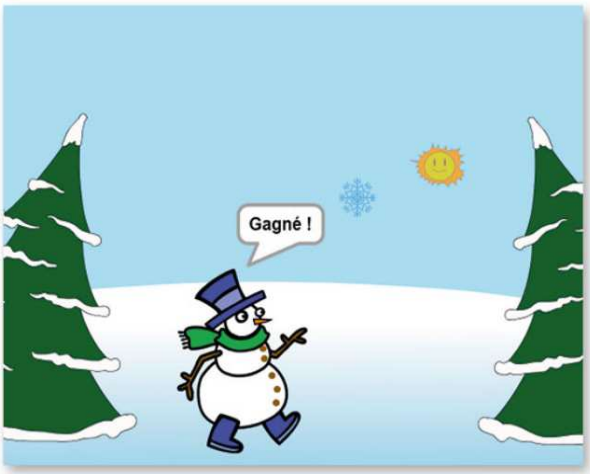
.....

- Pourquoi le jeu devient-il de plus en plus difficile ?

.....

.....

Étape 4 Arrêt du jeu



Lorsque *Snowman* atteint une taille à définir maximale (jeu gagné) ou minimale (jeu perdu), le jeu s'arrête et un message signale au joueur s'il a gagné ou perdu.

- Quelles conditions permettent l'arrêt du jeu ?

.....

.....

.....

.....

.....

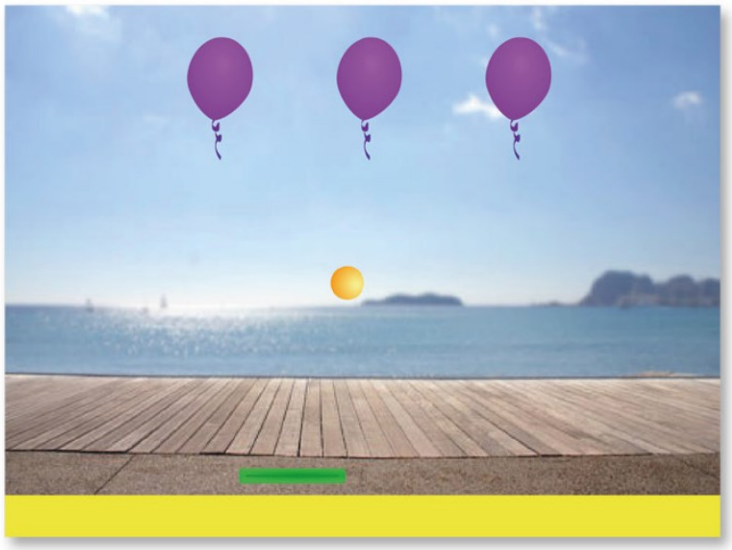
Pour aller plus loin :

- Afficher la durée du jeu et fixer une durée limite au-delà de laquelle le joueur perd s'il n'a pas encore atteint la taille maximale.
- Accélérer la chute des lutins *Snowflake* et *Sun* au fur et à mesure de l'avancée du jeu.

Éclate-ballons

Projet ceinture verte

- instructions conditionnelles
- boucles
- capteurs
- variables
- opérateurs



But du jeu : Le joueur doit faire rebondir la balle sur la raquette verte pour faire éclater les trois ballons. Il perd si la balle touche la bande jaune située sous la raquette.

Étape 1 L'arrière-plan et les lutins *Balloon*

Un rectangle jaune doit apparaitre en bas de la scène. Les trois ballons violets doivent être symétriques par rapport à l'axe des ordonnées.

- À quoi sert le rectangle jaune ? Pourquoi a-t-on choisi cette couleur ?

.....

.....

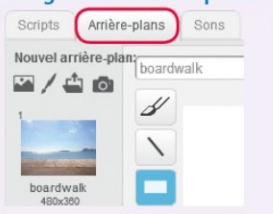
.....

.....

.....

Coup de pouce

Pour dessiner la bande jaune, tu peux utiliser la palette graphique dans l'onglet des arrière-plans.



- Que peut-on dire des coordonnées du centre des ballons de gauche et de droite ?

.....

Étape 2 Les lutins *Paddle* et *Ball*

Le lutin *Paddle* doit être dirigé horizontalement à l'aide de la souris, en bas de la scène, au-dessus du rectangle jaune. Le lutin *Ball* doit être orienté de façon aléatoire au début du jeu, avancer constamment et rebondir contre les bords de la scène.

- Quelle est la coordonnée de *Paddle* qui ne doit pas varier au cours de l'exécution du programme ?

.....

- Quel type de boucle permet d'assurer le déplacement de *Paddle* tout au long du jeu ?

.....

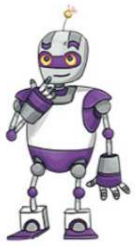
- Le lutin *Ball* devrait-il éviter certaines orientations au départ ? Pourquoi ?

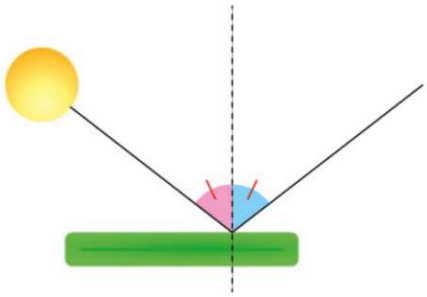
Étape 3 Interactions entre lutins

La balle doit rebondir de façon symétrique lorsqu'elle touche la raquette.
Le ballon doit disparaitre quand il est touché par la balle. Pour suggérer que le ballon éclate, on doit entendre un son.

Coup de pouce

Tu peux utiliser la variable **direction** qui contient, à chaque instant, l'orientation du lutin.





- Comment faut-il orienter la balle lorsqu'elle rebondit sur la raquette ?

Étape 4 Arrêt du jeu

Le jeu doit s'arrêter si la balle arrive dans la zone jaune ou si le joueur a éclaté les trois ballons.

- Quelle condition faut-il ajouter dans le script de *Ball* pour faire arrêter le jeu ?

.....

.....

.....

- Comment peut-on faire compter au programme le nombre de ballons éclatés ?

.....

.....

.....

Pour aller plus loin :

- Augmenter la vitesse de la balle après chaque ballon éclaté.
- Mettre en mouvement les ballons pour qu'ils se déplacent lentement sur la scène.
- Montrer un ballon seulement au début du jeu. Chaque ballon éclaté est remplacé par deux nouveaux ballons. Le joueur gagne lorsqu'il a éclaté dix ballons.

Chasse aux gobos

Projet ceinture noire

- boucles
- instructions conditionnelles
- variables
- clones

But du jeu : Le singe *Monkey* doit attraper les *Gobojaunes* tout en évitant d'être touché par les *Goboverts*. *Monkey* gagne 1 point lorsqu'il attrape un *Gobojaune*. Il perd la partie lorsqu'un *Gobovert* le touche.

Étape 1 *Monkey* et les *Gobojaunes*

Au début du jeu, *Monkey* doit apparaitre en bas à gauche de l'écran, puis il doit suivre continuellementle pointeur de la souris. Le jeu doit commencer lorsque le pointeur de la souris du joueur touche *Monkey*. Plusieurs *Gobojaunes* doivent alors apparaitre au centre de la scène, puis avancer indéfiniment dans une direction aléatoire en rebondissant sur les bords de la scène. Ils doivent disparaitre quand *Monkey* les touche, puis réapparaitre au centre de la scène.

Coup de pouce

Pour écrire le script, tu peux utiliser les commandes suivantes.

- Est-on obligé d'utiliser un bloc 

? Pourquoi ?

.....

.....

.....

- Quel type de commande doit-on utiliser pour démarrer le jeu lorsque le pointeur de la souris touche *Monkey* ?

.....

- Combien de clones doit-on créer pour avoir en tout dix *Gobojaunes* ?

.....

- Quelle commande doit-on utiliser pour positionner un lutin au centre de la scène ?

.....

Étape 2 Le score

Quand *Monkey* touche un *Gobojaune*, le joueur marque 1 point. Le *Gobojaune* touché doit alors disparaitre, puis réapparaitre au centre de la scène. Lorsque le score atteint 50 points, le joueur a gagné et le jeu s'arrête.

- Quelle doit être la valeur de la variable qui contient le score au début de la partie ?

.....

- Dans quel type de boucle doit-on mettre la commande qui permet de tester si le score a atteint 50 ? Justifier.

.....

.....

Étape 3 Les *Goboverts*

Au fur et à mesure de la partie, plusieurs *Goboverts* doivent apparaitre au centre de la scène puis avancer indéfiniment dans une direction aléatoire en rebondissant sur les bords de la scène. Quand *Monkey* touche un *Gobovert*, le jeu s'arrête et le score est annoncé.

- Comment peut-on modifier la vitesse de déplacement des *Goboverts* ?

.....

.....

.....

.....

- À quel moment peut-on faire apparaitre de nouveaux *Goboverts* ?

.....

.....

Pour aller plus loin :

- Accélérer le déplacement des *Gobos* à partir d'une certaine durée de jeu.
- Instaurer un temps limite pour le jeu.
- Enlever un certain nombre de points lorsque *Monkey* touche *Gobovert*, puis arrêter le jeu au troisième *Gobovert* touché.
- Créer des *Gobobonus* qui apportent plus de points une fois touchés.

Le jeu de Duck

Projet ceinture jaune

- variables
- instructions conditionnelles
- opérateurs

But du jeu : Quand on lance deux dés à six faces, on peut obtenir un total de deux à douze points en additionnant les valeurs obtenues. Le jeu proposé par *Duck* est-il équitable ?

Étape 1 Énoncer les règles du jeu

Duck doit commencer par proposer à *Beetle* un jeu, que ce dernier accepte. Le jeu doit afficher la valeur des deux dés de *Duck* et de *Beetle* ainsi que la somme de leurs dés respectifs.

- Quelle commande permet à *Beetle* d'attendre que *Duck* ait fini de parler pour lui répondre ?

.....

.....

- Combien de variables ont été créées ? Que représentent-elles ?

.....

.....

Étape 2 Lancer les dés

Chaque fois que l'utilisateur presse la touche *espace*, les lancers de dés doivent être simulés, et *Duck* et *Beetle* doivent annoncer leurs résultats.

- Quelle commande permet de choisir un nombre entier au hasard entre un et six ?

.....

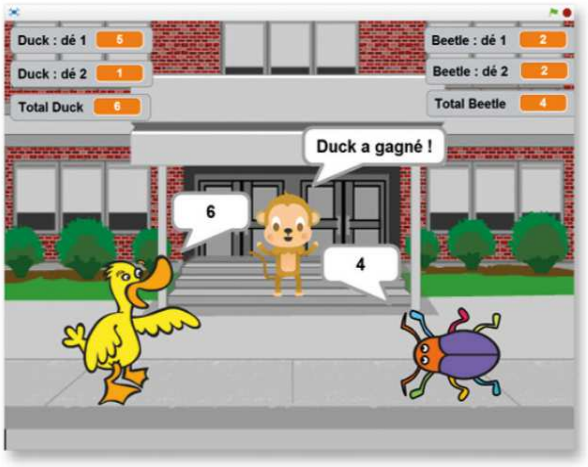
.....

- Comment faut-il compléter le bloc 

pour qu'il soit utilisé dans le script de *Duck* ?

Étape 3 Interpréter les résultats

Un troisième lutin doit annoncer si *Duck* ou *Beetle* a gagné, ou s'il y a égalité.



- Jouer plusieurs fois. Les règles proposées par *Duck* semblent-elles équitables ? Comment aurait-on pu le prévoir ?

- Proposer de nouvelles règles du jeu plus équitables.

Pour aller plus loin :

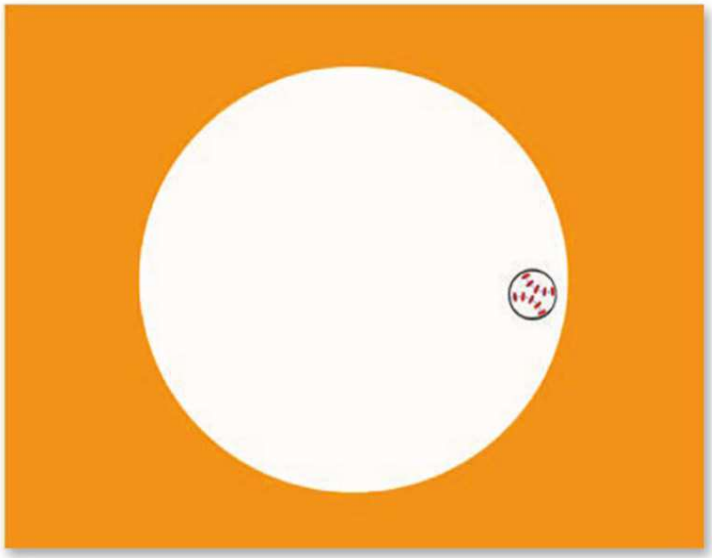
- Modifier le jeu pour que cette fois, *Duck* et *Beetle* lancent des dés à dix faces, appelés décaèdres. Trouver une règle du jeu équitable avec ces dés.
- Réaliser un programme qui simule mille parties et qui affiche combien de fois *Duck* et *Beetle* gagnent.

Prof : Manjoubi BIDI

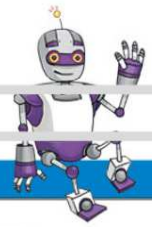
Dans le mille !

Projet ceinture verte

- variables
- boucles
- instructions conditionnelles
- capteurs
- opérateurs



But du jeu : La balle apparait aléatoirement sur la scène à chaque essai déclenché lorsque l'utilisateur presse la touche *espace*. On souhaite savoir combien de fois la balle apparait dans la cible blanche par rapport au nombre total d'essais.



Étape 1 Programmer les essais et le comptage

1

La scène doit faire apparaitre le lutin *Baseball* réduit à 50 % de sa taille et l'arrière-plan *light 494x372*.
L'utilisateurdoit pouvoir faire autant d'essais qu'il le souhaite en appuyant sur la touche *espace*. À chaque fois, la balle doit se placer aléatoirement sur la scène. Des variables permettent de mémoriser les résultats.

Coup de pouce

La scène est un rectangle de 480 pixels de large et de 360 pixels de haut.

- Quelles sont les valeurs minimales et maximales des coordonnées x et y de la balle ?

.....

- Combien de variables sont nécessaires ? Que représentent-elles ?

.....

.....

.....

.....

.....

- Lorsque la balle apparait en dehors de la cible blanche sans la toucher, comment les valeurs des variables changent-elles ?

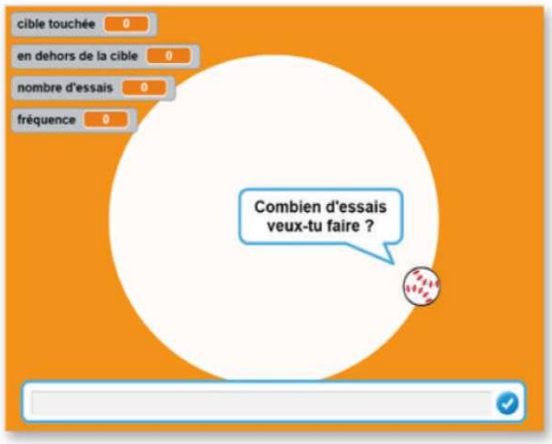
.....

.....

- La somme des variables qui représentent le nombre de fois où la balle apparait à l'intérieur et à l'extérieur de la cible est-elle égale au nombre d'essais ? Pourquoi ?

Étape 2 Faire un grand nombre d'essais

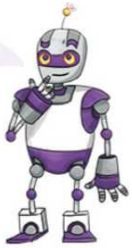
Pour que l'expérience soit plus rapide, on veut que l'utilisateur donne le nombre d'essais qu'il souhaite effectuer pour que le programme effectue automatiquement ces essais. Il doit ensuite afficher la fréquence de l'évènement « La balle a atteint la cible ».



Coup de pouce

Calcul de la fréquence f en pourcentage :

$$f = \frac{\text{nombre de fois où la cible a été touchée}}{\text{nombre d'essais}} \times 100$$



- Compléter le tableau de valeurs.

Nombre d'essais	50	100	500	1000
Nombre de fois où la cible a été touchée				

- Quelle est la fréquence de l'évènement « la balle a atteint la cible » en pourcentage pour 2 000 essais ?

.....

- Mettre la balle à 1 % de sa taille et refaire l'expérience avec un grand nombre d'essais. Que se passe-t-il ?

.....

- Y-a-t-il un lien entre le pourcentage de l'aire de la scène occupée par la cible blanche et la fréquence calculée par le programme ? Si oui, lequel ?

.....

.....

.....

.....

Coup de pouce

Tu peux augmenter la vitesse d'exécution du programme en activant le « Mode Turbo » dans le menu « Édition ».

Pour aller plus loin :

- Modifier l'arrière-plan et refaire la même expérience afin de connaitre les chances que le lutin atteigne une zone précise de la scène.
- Dessiner un arrière-plan composé d'un triangle équilatéral coloré puis utiliser cette expérience pour déterminer une valeur approchée de son aire.

Habillage à l'aveugle

Projet ceinture noire

- variables
- boucles
- instructions conditionnelles
- opérateurs

Chaquematin, le chat choisit les yeux fermés un nœud papillon et une paire de lunettes.
Il affirme passer un tiers de l'année tout en rouge, est-ce vrai ?

But du jeu : Savoir s'il a raison.

Étape 1 Habiller le chat

Le chat possède trois nœuds papillons et trois paires de lunettes de trois couleurs : rouge, vert et bleu.
L'utilisateur doit pouvoir faire changer la tenue du chat à chaque fois qu'il clique sur le bouton Changer de tenue.

- Quelle commande permet au chat d'avoir des lunettes vertes ?

.....

.....

.....

Coup de pouce

- Le bouton, le nœud papillon et les lunettes sont des lutins dont les costumes ont été modifiés.
- Pour dupliquer un costume, tu peux faire un clic droit sur le costume :

- Pour changer la couleur d'un costume, tu peux utiliser l'outil .
- Pour insérer du texte, tu peux utiliser l'outil T.

Étape 2 Programmer le choix d'une tenue pour un jour

Lorsque l'utilisateur clique sur le bouton Changer de tenue, la paire de lunettes et le nœud papillon doivent prendre une couleur aléatoire parmi rouge, vert et bleu.

- Dans la zone de scripts de quel lutin doit-on mettre les commandes suivantes ?

Coup de pouce

Tu peux créer deux variables *N* (pour numéro de nœud) et *L* (pour numéro de lunettes) qui prendront aléatoirement une valeur comprise entre 1 et 3.

.....

- Pour le nœud papillon, combien de tests « Si *N* = … » doit-on faire ? Pourquoi ?

.....

.....

- Faire l'expérience sur deux semaines et noter dans le tableau les couleurs du nœud papillon et des lunettes du chat pour chaque jour. Noter R pour rouge, B pour bleu et V pour vert.

Jour	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Couleur du nœud papillon														
Couleur des lunettes														

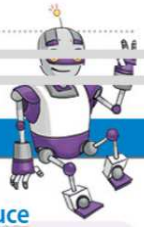
- Sur ces deux semaines, combien de fois le chat a-t-il porté une tenue rouge, c'est-à-dire un nœud papillon et des lunettes rouges ?

.....

- Cela permet-il de déterminer si l'affirmation du chat est vraie ? Pourquoi ?

.....

.....



Étape 3 Compter le nombre de tenues monochromes sur un an

Le programme évolue : lorsque l'utilisateur clique sur le bouton Changer de tenue, le programme doit faire directement l'expérience sur toute une année.
À la fin, l'utilisateur voit s'afficher le nombre de tenues rouges que le chat a portées.

Coup de pouce
Créer une variable pour compter le nombre de tenues rouges.

- Combien de fois le programme doit-il changer la tenue du chat ?

.....

- Répéter l'expérience afin de simuler trois années de suite. Combien de tenues rouges le chat a-t-il portées ?

.....

- L'affirmation du chat est-elle vraie ?

.....

.....

.....

Coup de pouce
Tu peux augmenter la vitesse d'exécution du programme en activant le « Mode Turbo » dans le menu « Édition ».

Pour aller plus loin :
- Trouver la fréquence d'apparition d'une tenue bicolore.
- Ajouter un chapeau au chat, avec plusieurs couleurs possibles, et refaire une expérience.