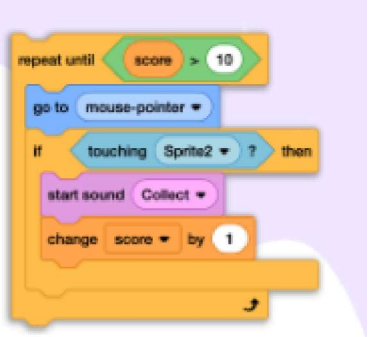


Créer des projets interactifs avec des i nstructions conditionnelles

Avez-vous déjà eu envie de créer un programme Scratch interactif ou offrant plusieurs résultats ? Certa ins programmes Scratch sont statiques : le résultat est fixe et la même chose se produit à chaque fois. C ertains sont dynamiques : ils sont capables d'agir ou de changer à chaque fois qu'ils sont exécutés. Afin de créer des programmes dynamiques, le programmeur peut utiliser des blocs d'instructions conditionn elles pour donner des instructions sur la façon dont le projet doit réagir dans différentes circonstances.

Dans ce guide, vous trouverez :

Que sont les instructions conditionnelles ● Jusqu'à vrai ou faux ● Si vrai ou faux ● Boucles conditionnelles ● Exemples d e conditions

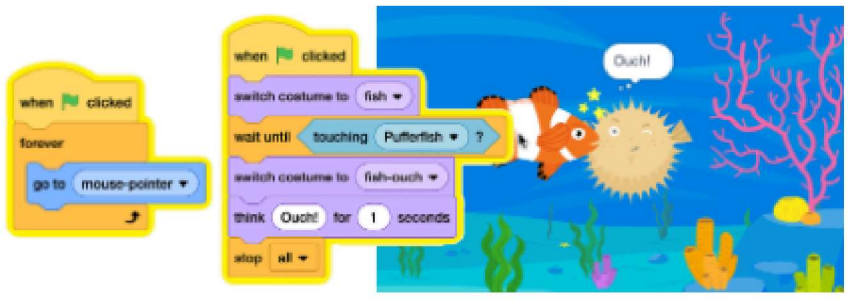
Débogage des instructions conditionnelles ● Voies/solutions multiples ● Pratique débranc hée avec les instructions conditionnelles

Que sont les instructions conditionnelles

Imaginez un projet où un poisson peut être contrôlé par la souris de l'utilisateur (voir exemple ici). Le sc ript de Thefish dit : pour toujours, allez au pointeur de la souris. Et si vous vouliez qu'une action unique ait lieu si le poisson touche un autre sprite, comme un poisson-globe ?

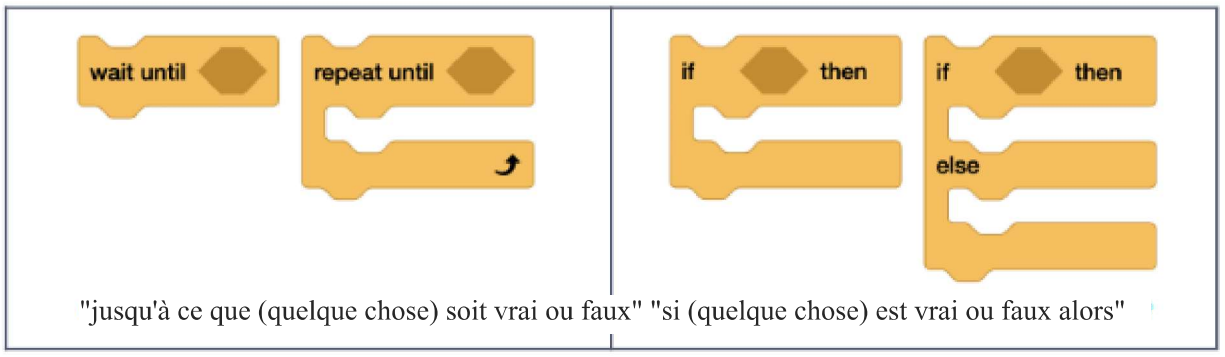
Nous pouvons créer un script qui utilise un bloc d'instructions conditionnelles « attendre » et indique au pro gramme d'attendre que les sprites se touchent. Ensuite, une fois touché, nous pouvons programmer une actio n comme changer de costume et dire « Aïe ! » Désormais, le programme est dynamique et interactif !

Si l'utilisateur ne déplace jamais le poisson sur le poisson-globe, rien ne se passe. ● S'il dépl ace le poisson au-dessus du poisson-globe, il peut voir une animation amusante !



Une instruction conditionnelle peut être vraie ou fausse. Dans ce cas, si l es esprits se touchent, c'est vrai. S'ils ne se touchent pas, c'est faux.

Dans la catégorie Contrôle de Scratch, vous trouverez un certain nombre de blocs d'instructions condition nelles qui vous permettent d'ajuster votre programme en fonction de conditions spécifiques. Dans Scratch, vous rencontrerez deux types d'instructions conditionnelles :

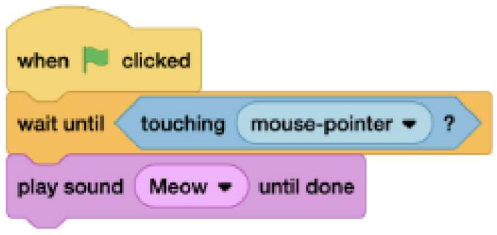


"jusqu'à ce que (quelque chose) soit vrai ou faux" "si (quelque chose) est vrai ou faux alors"

Explorons ces blocs d'instructions conditionnelles, pratiquons avec quelques exemples et apprenons à cr éer des programmes dynamiques et interactifs !

Jusqu'à ce que ce soit vrai ou faux

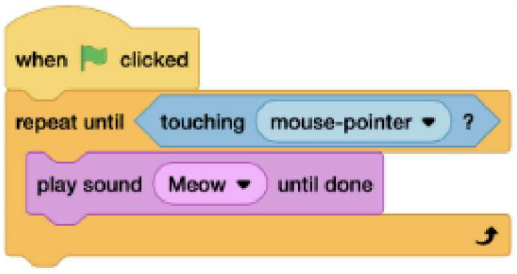
Dans le jeu de poisson ci-dessus, nous avons utilisé un bloc « jusqu'à ce que (quelque chose) soit vrai ou faux » pour déclencher une action. Les deux scripts ci-dessous utilisent « attendre » et « répéter jusqu'à » pour effectuer des actions. Regardons de plus près pour voir la différence :



"Attendez jusqu'à"

Commencez par un script simple comme : lorsque v ous cliquez sur le drapeau vert, attendez qu'une cert aine condition soit vraie et jouez un son. Dans ce ca s, nous pourrions regarder dans la catégorie de déte ction et choisir une condition comme « toucher le p ointeur de la souris ».

Pour tester ce script et voir comment il fonctionn e, cliquez sur le drapeau vert, attendez, puis passe z votre souris sur votre sprite. Votre son ne sera j oué que lorsque la souris touche le sprite.



"Répéter jusqu'à"

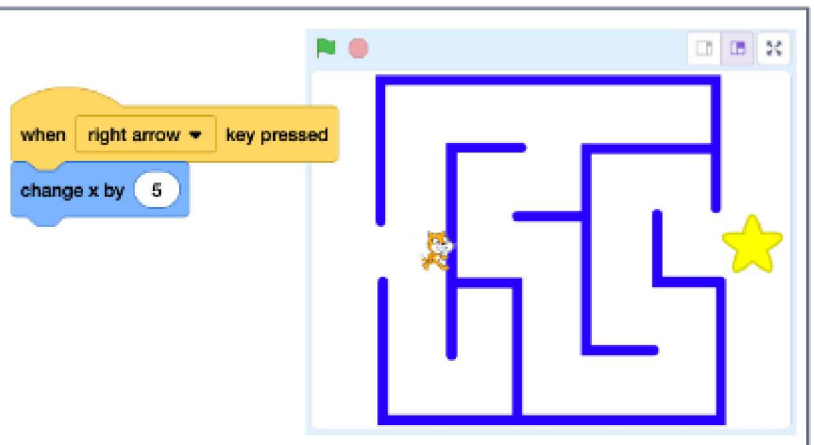
Ajustez votre script pour dire : lorsque vous cliq uez sur le drapeau vert, répétez la lecture d'un s on jusqu'à ce qu'une certaine condition soit vrai e. (Un son court est préférable pour ce test.) Util isons la même condition « toucher le pointeur d e la souris ».

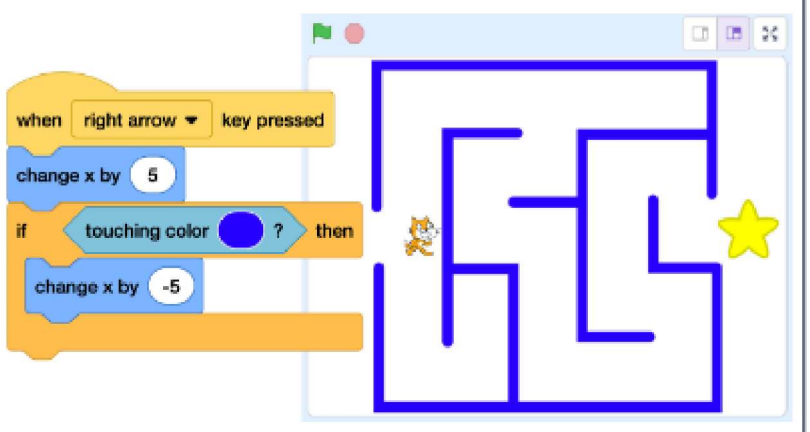
Pour tester ce script et voir la différence, cliqu ez sur le drapeau vert, attendez, puis passez vo tre souris sur votre sprite. Le son est joué à plu sieurs reprises et ne s'arrête qu'une fois que vot re souris touche le sprite.

Si vrai ou faux

Une autre version des instructions conditionnelles dans Scratch vérifie « si (quelque chose) est vrai ou faux ». Par exemple, disons que vous souhaitez créer un jeu de labyrinthe (voir exemple ici) dans lequel le joueur contrôle le sprite avec les touches fléchées du clavier :

<table>
<tr>
<td>Code initial

Lorsqu'une touche est enfoncée, le sprite est codé pour se déplacer d'un certain nombre de pas. Mais vous devez empêcher le lutin de traverser les murs. Qu’est-ce qui peut rendre cela possible ? Déclarations conditionnelles !</td>
<td> 
</td>
</tr>
<tr>
<td>Ajouter une instruction « Si alors »

Nous pouvons utiliser la couleur des murs du labyrinthe comme condition pour affecter le comportement du sprite. Ajoutez un bloc « si alors » au bas de la séquence. Utilisez le bloc de détection « Toucher la couleur » comme condition et sélectionnez la couleur des murs. Que se passe-t-il s’il touche les murs ? Si le sprite se déplace du même nombre de pixels dans la direction opposée, il semblera que le sprite n'a pas bougé du tout et a été arrêté par le mur.</td>
<td> 
</td>
</tr>
</table>

Testez ce code de labyrinthe, puis ajoutez des scripts supplémentaires pour déplacer le sprite vers la gauche, le haut et le bas.

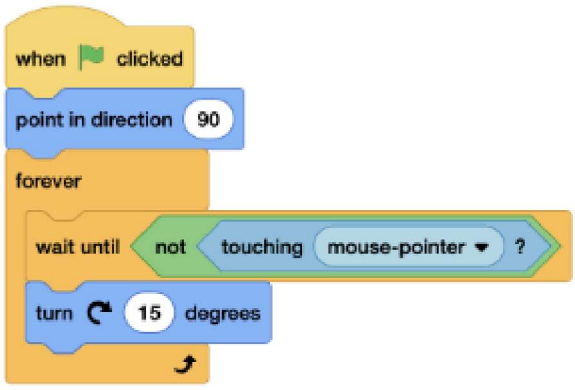
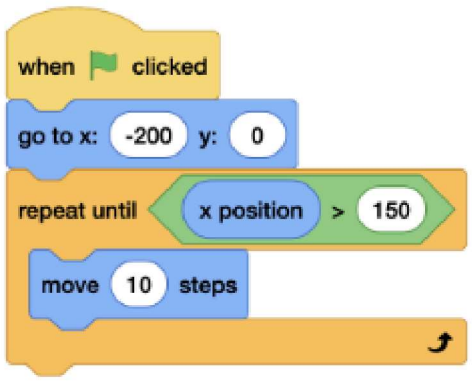
Comment pourriez-vous utiliser un bloc « si alors » ou « si alors sinon » pour programmer un message gagnant, un son ou une célébration animée lorsque le sprite atteint la fin du labyrinthe ? Comment pourriez-vous utiliser des instructions conditionnelles pour programmer des obstacles à éviter ou des objets à collecter pour gagner des points ?

Boucles conditionnelles

Que se passe-t-il si vous souhaitez que le programme vérifie continuellement si la condition est vraie ou faus se («écouter» continuellement) et effectue des actions encore et encore ? Vous devrez placer la séquence dan s une boucle pour créer une boucle conditionnelle. Comparez la différence entre ces deux scripts :

	
Dans cet exemple, une fois la condition remplie et le code exécuté, le programme s'arrête (vous pouvez savoir que le script a arrêté de s'exécuter car il n'est plus mis en surbrillance). It only“liste ns” once for the condition.	Dans cet exemple, le programme « écoute » en per manence pour voir si la condition est vraie. Le scri pt à l'intérieur de la boucle éternelle redémarre un e fois que la condition est remplie et que le code e st exécuté. Il vérifie à nouveau si la condition est v raie/faux.

Exemples de conditions

	
Opérateur « non » Et si nous voulons que quelque chose se produise si quelque chose d'autre ne se p roduit pas (comme si le pointeur de la souris ne t ouche pas le sprite) ? Dans la catégorie Opérateur s, recherchez un opérateur « non ». Testez la pile de code ci-dessus en plaçant le pointeur de la sou ris sur et hors du sprite. Quelle différence remarq uez-vous ?	Opérateurs de comparaison Les opérateurs de co mparaison comparent les valeurs de deux élément s de données et déterminent si l'équation est vraie ou fausse. Cette instruction conditionnelle ci-dess us indique que le sprite doit se déplacer de 10 pas à la fois jusqu'à ce que la position x soit supérieur e à 150. Testez cette pile de codes et voyez le résu ltat.

Il existe une grande variété de conditions dans Scratch parmi lesquelles vous pouvez choisir pour com pléter votre déclaration conditionnelle. Les blocs qui rapportent des valeurs « vraies » ou « fausses » s ont appelés blocs booléens et vous pouvez les identifier par leur forme hexagonale allongée. Voici que lques exemples de conditions que vous pourriez utiliser :

- actions de l'utilisateur, telles que l'appui sur certaines touches du clavier ou le positionnement ou le clic de la souris

- interactions entre sprites (toucher un autre sprite), comparaisons (distance entre les sprites) et toucher les couleurs des sprites ou des arrière-plans

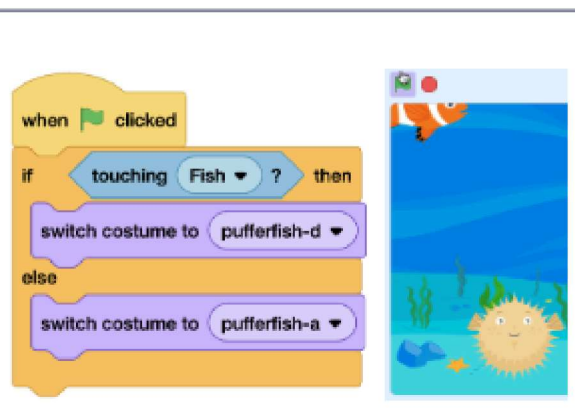
- données saisies par les utilisateurs, données stockées dans des varia bles et des listes, ou données stockées dans des blocs de rapporteur

Débogage des instructions conditionnelles

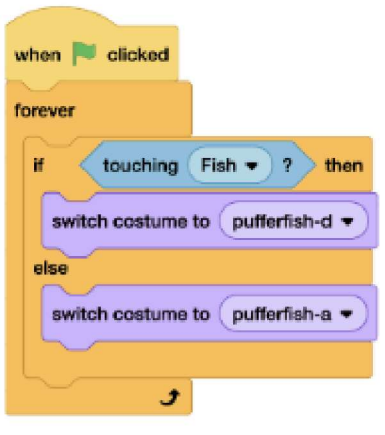
Lorsque vous créez un programme, vous souhaiterez peut-être effectuer des tests utilisateur en demanda nt à quelques amis ou membres de votre famille de l'essayer. Tout fonctionne-t-il comme prévu ou deve z-vous effectuer un débogage pour corriger des erreurs ou des comportements inattendus ?

Débogage : pourquoi ne vérifie-t-il pas ?

Si vous n'avez besoin que du programme pour v érifier votre condition une seule fois, vous n'ave z pas besoin de placer votre instruction conditio nnelle à l'intérieur d'une boucle. Mais si vous vo ulez que le programme vérifie continuellement s i la condition est vraie ou fausse (écoutez contin uellement) et effectue des actions encore et enco re, vous devrez placer la séquence de codes à l'i ntérieur d'une boucle pour créer une boucle cond itionnelle.



Par exemple, dans le premier exemple à droit e, lorsque je démarre le programme fish et que je clique sur le drapeau vert, le poisson et le p oisson-globe ne se touchent pas, donc le progr amme arrête le script car la condition a été vér ifiée et était fausse.



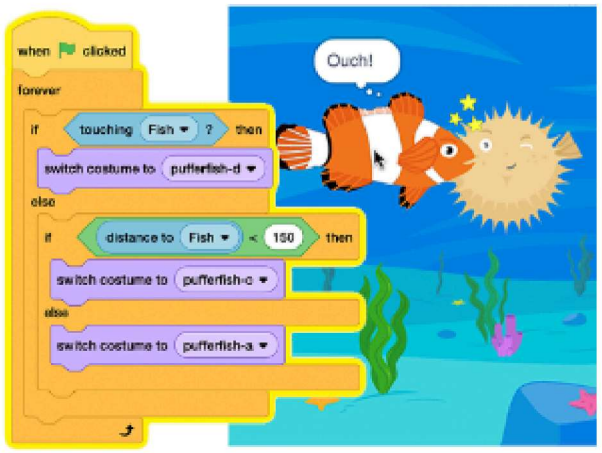
Dans le deuxième exemple, le programme vérifi e en permanence si la condition est vraie ou fau sse et exécutera le code sous « ifthen » lorsqu'el le est vraie ou sous « else » lorsqu'elle est fauss e.

Débogage : ordre séquentiel

Supposons que je veuille ajouter du code au s prite du poisson-globe dans le jeu de poisson, afin que le joueur voie différents costumes da ns différentes circonstances.

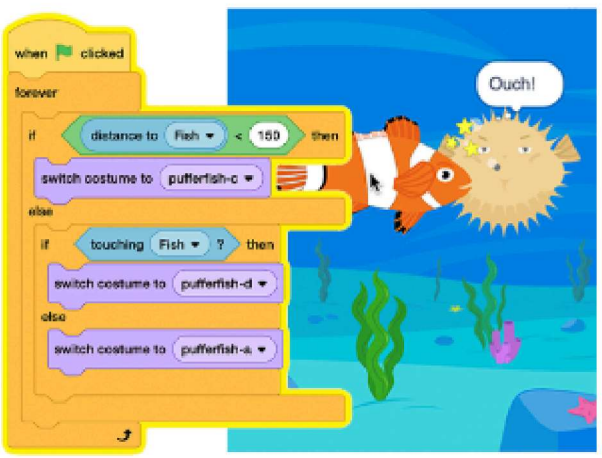
Dans le premier exemple à droite, j'ai imb riqué deux instructions « if then else » :

> Tout d'abord, le programme vérifie si les sp rites se touchent, et si c'est vrai, alors il affi che le costume d. ● Sinon, si le contact est f aux, il vérifie si la distance entre les sprites est proche, et si c'est le cas, affiche le costu me c.

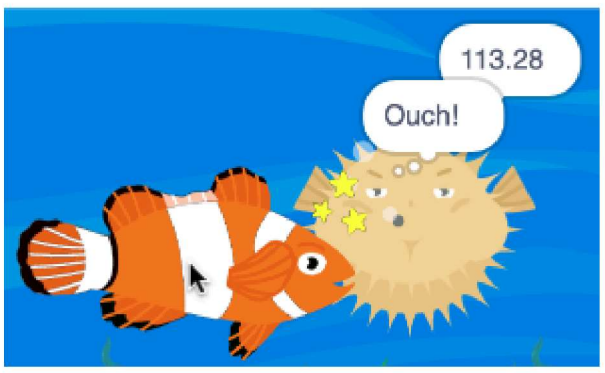


Et si l'ordre des instructions imbriquées « if the n else » était différent ? Dans le deuxième exem ple :

> Tout d'abord, le programme vérifie si la dist ance entre les sprites est proche, et si c'est le cas, il affiche le costume c. ● Sinon, si la dis tance est supérieure à 150 pixels, il vérifie si les sprites se touchent, et si cela est vrai, il a ffiche le costume d. ● Sinon, il affiche le co stume initial a.



Dans le deuxième exemple, on ne voit jamais le poisson-globe clignotant (costume d) lorsq ue les esprits se touchent. Pourquoi?



Dans l'ordre séquentiel du deuxième exemple, le programme vérifie d'abord si la distance ent re les sprites est inférieure à 150. Puisque cela est également vrai lorsque les sprites se touche nt, le programme ne passe pas à la vérification du contact.

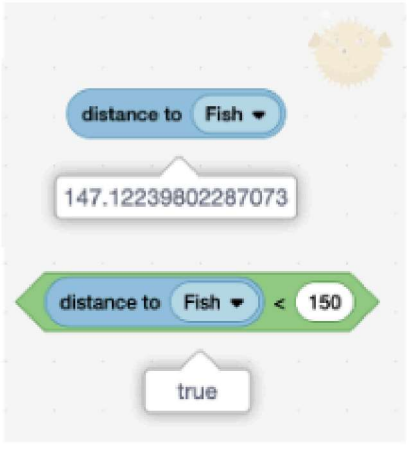
Débogage : expressions mathématiques

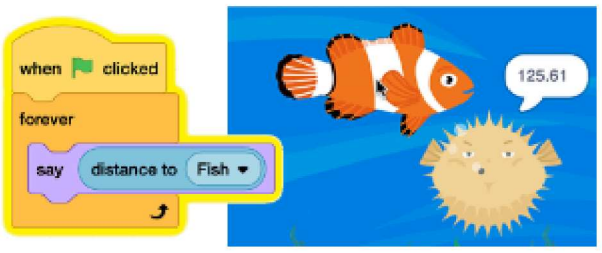
Pendant que je travaille sur mon code de poisso n-globe, comment puis-je déterminer la distance que je souhaite saisir dans mon opérateur de co mparaison qui vérifie la distance ? Il peut être di fficile de juger visuellement la distance entre les pixels.

Ce rapporteur « distance jusqu'au poisson » ind iquera la distance entre les points centraux des deux sprites. Cliquez sur le bloc reporter dans l a zone de script pour voir la valeur.

Maintenant, mettez « distance jusqu'à » dans un bloc d'opérateur comme supérieur à ou inférieu r à, rapprochez et séparez les sprites et cliquez s ur la condition dans la zone de script pour voir si elle est vraie ou fausse.

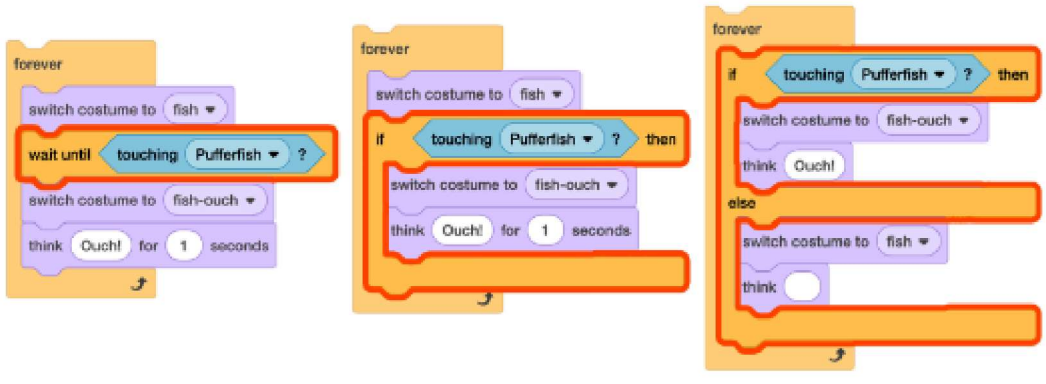
Si vous souhaitez bricoler et voir rapidement q uelle est la distance mesurée lorsque le poisson est dans des positions indifférentes, vous pouve z également demander au poisson-globe de dire continuellement la « distance au poisson » lors que vous déplacez le poisson.





Plusieurs voies/solutions

Il existe souvent plusieurs solutions/plusieurs façons de coder un programme pour obtenir un résultat similaire. Par exemple, si nous revenons au code du poisson dans le jeu de poisson, disons que je souhaite ajuster le code pour ajouter une boucle conditionnelle afin que le poisson et le poisson-globe puissent interagir plus d'une fois.



Notez que les trois solutions ci-dessus utilisent des blocs similaires, mais l'une utilise « attendre jusqu'à », une autre utilise « si alors » et la troisième utilise une instruction conditionnelle « si alors sinon ». Regard ez, recréez et bricolez ces trois scripts. Qu'est-ce qui est pareil et qu'est-ce qui est différent ?

Pratique débranchée avec les instructions conditionnelles

Explorez les blocs conditionnels if/then et while/jusqu'à ce que nous utilisions notre activité de jeu débranché e Simon Says Conditional Statements. Une personne joue le rôle de programmeur tandis que les participants, en tant qu'ordinateurs, doivent déterminer si les déclarations sont vraies ou fausses, puis agir en conséquence.

Consultez nos cartes de codage associées : Cartes de codage des instructions conditionnelles

Regardez nos vidéos de ressources complémentaires ici pour en savoir plus :

Déclarations conditionnelles : réaliser des projets interactifs (partie 1) | TutorielInstructions conditionnelles : imbrication, débogage et au-delà (partie 2) | Tutoriel

Astuce : Si vous souhaitez traduire ce guide, cliquez ici pour faire une copie de ce document Google.