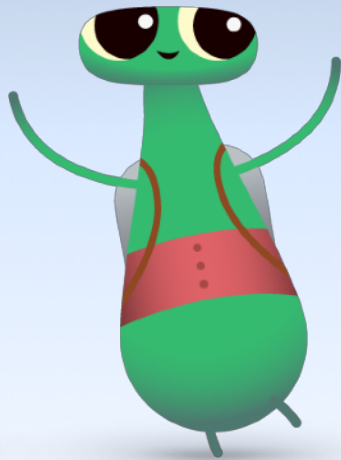




Swift Coding Club



Swift Playgrounds

Activités de programmation

Thèmes

- | | | |
|----------|---|----|
| 1 | Penser comme un ordinateur :
commandes et séquences | 3 |
| 2 | Penser efficacement :
fonctions et un soupçon de boucles | 5 |
| 3 | Penser logiquement :
code conditionnel | 7 |
| 4 | Penser encore et encore :
boucles while | 9 |
| 5 | Penser la même idée :
algorithmes | 11 |
| 6 | Penser comme un newsbot :
variables | 13 |
| 7 | Penser comme un architecte :
types | 15 |

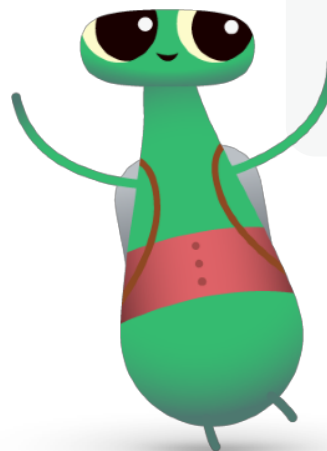
Bienvenue au Club de programmation Swift !

Ces activités de programmation abordent des concepts de programmation fondamentaux pour accompagner les activités de conception d'app. Elles vous permettront non seulement de renforcer vos compétences en matière de programmation, mais aussi de mieux comprendre le fonctionnement des apps. Ainsi, vous serez mieux armé pour concevoir des apps plus abouties.

Pour chaque thème, vous découvrirez un concept de programmation spécifique lors d'une brève activité d'introduction, puis vous mettrez ce concept en application pour résoudre des puzzles dans Swift Playgrounds.

Vous trouverez également des activités « Pour aller plus loin », afin d'enrichir vos connaissances sur le concept en question. Ces activités sont facultatives : vous pouvez choisir d'en faire une seule ou les deux, ou de n'en faire aucune. Notez que certaines d'entre elles nécessitent un autre appareil, tel qu'un robot ou un drone. Si vous possédez de tels appareils, sachez qu'ils constituent un moyen idéal d'appliquer les connaissances acquises.

Amusez-vous bien !



Penser comme un ordinateur : commandes et séquences



Introduction (15 minutes)

Faites équipe avec un camarade. L'un de vous sera le « directeur » ; l'autre, l'exécutant. Le directeur formulera une idée d'action que l'exécutant devra effectuer. Il peut s'agir de dessiner un smiley au tableau ou de faire cinq jumping jack. Sans lui présenter la tâche dans son ensemble, le directeur devra fournir à l'exécutant des instructions détaillées pour chacune des étapes. L'exécutant a-t-il accompli la tâche comme prévu ?

Le directeur a donné à l'exécutant des commandes organisées selon une certaine séquence. C'est exactement ce que vous devez faire lorsque vous écrivez du code.



Regardez cette [vidéo](#) pour en savoir plus sur les commandes, et [celle-ci](#) pour comprendre ce qu'est le débogage.

Maintenant, revoyez vos instructions, surtout si l'exécutant n'a pas réussi à effectuer la tâche telle qu'elle était prévue. Manquait-il une étape dans les instructions ? Ou, si vous aviez modifié l'ordre de certaines étapes, les instructions auraient-elles été plus claires ? C'est ce que l'on appelle le débogage (ou « débogage »). Les programmeurs doivent souvent faire du débogage pour corriger et améliorer leur code.

Mise en pratique (30 minutes)

Maintenant, utilisez Swift Playgrounds : Apprendre à coder 1 pour résoudre les puzzles cochés en vert dans la liste de droite.

Pensez-y : les commandes que vous avez utilisées dans l'app ressemblaient-elles aux instructions fournies par le directeur ?

Commandes

Introduction	✓
Création de commandes	✓
Ajout d'une nouvelle commande	✓
Activation d'un interrupteur	✓
Utiliser le téléporteur	✓
Recherche et correction de bugs	✓
Entraînement au débogage	✓
Le chemin le plus court	✓

Commande : action particulière que l'ordinateur doit effectuer.

Séquence : l'ordre d'envoi des commandes.

Débogage : processus consistant à identifier et corriger cette erreur.



Penser comme un ordinateur : commandes et séquences

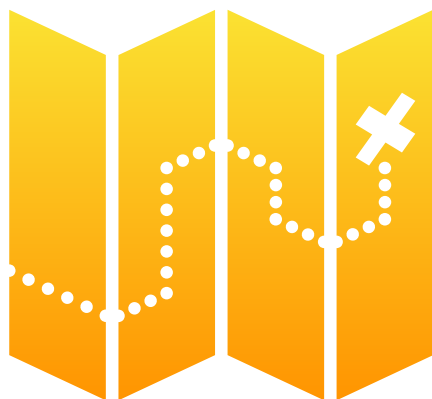
Pour aller plus loin

1

Cache-cache (1 séance)

1. Cachez un petit objet dans la pièce ou à proximité.
2. Postez-vous à un endroit précis et utilisez la caméra de l'iPad pour vous enregistrer en train de donner des instructions permettant de trouver l'objet. Les instructions doivent partir de l'endroit où vous vous tenez.
3. Maintenant, échangez votre vidéo avec celle d'un autre élève. Regardez sa vidéo pour trouver l'objet qu'il a caché. L'avez-vous trouvé ?

Pensez-y : en quoi les instructions pourraient-elles être améliorées ? Les instructions ont-elles besoin d'être déboguées ? Si oui, comment ?



Dash (1 séance)

1. Si vous disposez de robots Dash de Wonder Workshop, passez à la leçon Dash dans Swift Playgrounds.



2. Utilisez des commandes pour guider Dash tout au long de la journée de courses.
3. Le trajet de Get to the Race peut être un peu difficile. Comparez votre code à celui d'autres élèves. Si nécessaire, revoyez vos codes respectifs et procédez ensemble au débogage.
4. Voyez à quel stade vous parvenez. Vous pourrez toujours revenir à cette leçon une fois que vous aurez découvert d'autres concepts de programmation.
5. Dès que vous serez prêt, créez votre propre histoire, dont Dash sera le personnage principal.

Pensez-y : quels capteurs de Dash avez-vous utilisés pour vous aider à raconter votre histoire ? En quoi ces capteurs l'ont-ils enrichie ?

Penser efficacement : fonctions et boucles


2



Introduction (5 minutes)

Pensez à quelques mouvements de danse. Décrivez-vous mutuellement ces mouvements ou, mieux, faites-les exécuter les uns par les autres. A-t-il été facile de les décrire ?

En programmation, il est parfois plus facile de combiner des commandes existantes pour créer de nouveaux comportements. C'est ce que l'on appelle la composition. En nommant un nouveau comportement afin de pouvoir le réutiliser par la suite, vous créez une fonction. Lorsque vous indiquez à un programme d'exécuter une fonction, vous « appelez » celle-ci. Donc lorsque l'on dit « danse la Macarena », on appelle la fonction « Macarena ».

 Regardez cette [vidéo](#) sur les fonctions et les boucles.

Mise en pratique (40 minutes)

Maintenant, utilisez Swift Playgrounds : Apprendre à coder 1 pour résoudre les puzzles cochés en vert dans la liste de droite.

Pensez-y : dans un puzzle donné, combien de mouvements le personnage a-t-il effectués ? Et combien de commandes avez-vous écrites ? Quand et pourquoi faut-il créer des fonctions et des boucles ?

Fonctions

Introduction	✓
Composition	✓
Création d'une nouvelle fonction	✓
Collecte, active, recommence	
Plateau garni	
Imbrication de patterns	✓
Marches et escaliers	✓
Chasse au trésor	

Boucles For

Introduction	✓
Utilisation des boucles	✓
Boucler tous les côtés	✓
Jusqu'au bord et retour	
Sauteur en boucle	
Explorer toutes les branches	
Ferme aux gemmes	
Balayage en quatre mouvements	

Fonction : ensemble de commandes groupées et dotées d'un nom.

Boucle for : exécute un bloc de code de manière répétée un certain nombre de fois.



Penser efficacement : fonctions et boucles

Pour aller plus loin

2

Machine à motifs (1 séance)

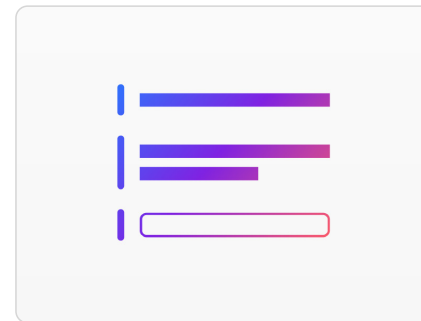
1. Créez un motif (ou « pattern », ou « schéma ») à l'aide d'une app de dessin telle que Art Set ou Pages, en y intégrant divers objets, figures et couleurs. Le motif peut être aussi long que vous le souhaitez.
2. Dans l'app, décrivez votre motif 20 fois à l'aide de mots : par exemple, « rouge, jaune, bleu ; rouge, jaune, bleu... », etc.
3. Donnez un nom à la partie du motif qui se répète (par exemple : rouge, jaune, bleu = Couleurs primaires), puis décrivez à nouveau le motif en utilisant uniquement ce nom.
4. Quelle quantité de travail ou combien d'étapes vous êtes-vous épargnées pour décrire le motif par écrit ?
5. Combien de fois l'expression Couleurs primaires s'est-elle répétée ? Maintenant, décrivez votre motif en une étape. Vous avez écrit une boucle for !

Pensez-y : quel est le lien entre cette activité et la programmation ?



Entretien avec un robot (1 séance)

1. Partez du Point de départ Réponses de Swift Playgrounds.



2. Sur la page Texte, « show » (montrer) et « ask » (demander) sont des fonctions. Touchez Exécuter mon code et inscrivez votre nom, puis touchez Envoyer pour voir ce qui se passe. Les fonctions peuvent avoir un résultat, qui est ce que l'on voit dans la vue en direct.
3. Sur la page Types, explorez différentes fonctions « show » et « ask ».
4. Faites équipe avec un camarade du club et écrivez chacun une série de fonctions « show » et « ask », que l'autre devra compléter.
5. Utilisez les résultats de vos fonctions pour rédiger une fiction, une interview ou une brève biographie.

Pensez-y : que se passerait-il si vous écriviez vos fonctions « show » et « ask » dans un ordre différent ? En quoi cela affecterait-il votre fiction ou votre interview ?

Penser logiquement : code conditionnel


3



Introduction (10 minutes)

En groupe, faites quelques tours du jeu J'espionne. Dans ce jeu, un espion choisit un objet, puis ne le décrit qu'en partie. Les autres doivent observer leur environnement et deviner ce qu'a vu l'espion. L'élève qui trouve la réponse devient l'espion.

Quelles décisions avez-vous dû prendre ? Quel mode de réflexion avez-vous adopté pour essayer d'identifier ce qu'a vu l'espion ? Pour jouer à J'espionne, vous avez réfléchi de manière conditionnelle.

 Regardez cette [vidéo](#) pour en savoir plus sur le code conditionnel.

Mise en pratique (35 minutes)

Maintenant, utilisez Swift Playgrounds : Apprendre à coder 1 pour résoudre les puzzles cochés en vert dans la liste de droite.

Pensez-y : quels types de décisions votre code a-t-il pris à l'aide des instructions if ? Comment avez-vous combiné les boucles for et les instructions if ? Pourquoi ?

Condition : quelque chose que vous testez et qui renvoie vrai ou faux.

Code conditionnel : bloc de code qui n'est exécuté que si quelque chose est vrai.

Booléen : valeur qui ne peut être que vraie ou fausse.

Opérateur logique : symbole ou mot comme « et » (« and »), « ou » (« or ») et « pas » (« not »).



Code conditionnel

Introduction	✓
Recherche d'interrupteurs	✓
Utilisation d'else if	✓
Code conditionnel en boucle	✓
Ascension conditionnelle	
Des fonctions plus intelligentes	✓
Dans la boîte	
Arbre de décision	

Opérateurs logiques

Introduction	✓
Utilisation de l'opérateur NOT	✓
Spirale de NOT	
Vérifier ceci AND cela	✓
Vérifier ceci OR cela	✓
Labyrinthe logique	

Penser logiquement : code conditionnel

Pour aller plus loin

3

Chasse au trésor (1 séance)

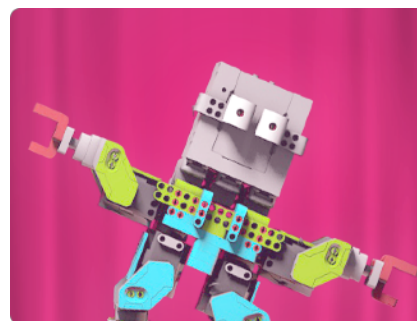
1. Chaque élève doit écrire deux conditions sur des morceaux de papier séparés : par exemple, « Est un rectangle » ou « Commence par la lettre C ». Placez ensuite les morceaux dans un chapeau.
2. Par petits groupes, piochez deux questions dans le chapeau et prenez trois à cinq photos d'objets qui respectent chaque condition.
3. Créez un album photo à l'aide d'une app de présentation telle que Keynote, en créant une section pour chaque condition. À ce stade, ne libellez pas les conditions.
4. Présentez votre album photo aux membres d'un autre groupe pour voir s'ils devinent les conditions. S'ils trouvent la solution, ajoutez l'instruction conditionnelle – par exemple : « si bleu, prendre en photo » – à la page de l'album.

Pensez-y : dans certains cas, a-t-il été difficile de déterminer si une photo répondait aux conditions énoncées ? Comment un ordinateur traiterai-il ces cas ?



Danses de MeeBot (1 séance)

1. Si vous disposez d'un robot MeeBot d'UBTECH, téléchargez la leçon Danses de MeeBot de Swift Playgrounds.



2. Parcourez la leçon et déterminez la façon dont vous pouvez mettre en œuvre des fonctions, des boucles for et du code conditionnel pour faire danser votre robot MeeBot.
3. Créez votre propre chorégraphie. Vous pouvez même choisir votre musique, sur la dernière page.

Pensez-y : comment avez-vous utilisé le code pour refléter le tempo de la musique ?

Penser encore et encore : boucles while

4




Introduction (5 minutes)

Dans le thème 2, vous avez découvert les fonctions et les boucles for. Quelle danse avez-vous interprétée ensemble ? Comment utiliseriez-vous des boucles for pour écrire la fonction de la danse ?

Maintenant, imaginez que vous vouliez interpréter cette danse à la fête de l'école. Comment sauriez-vous quand vous arrêter de danser à la fin du morceau ?

Vous utiliseriez une instruction conditionnelle : si le morceau joue, danse. Ou, plus précisément, une boucle while : tant que (while) le morceau joue, danse.

Cette option est différente d'une boucle for, qui indique à l'ordinateur d'exécuter un bloc de code un certain nombre de fois, comme dans « fais la ronde 10 fois ». Une boucle while indique à l'ordinateur d'exécuter un bloc de code jusqu'à ce qu'un événement survienne. Comme dans fais la ronde jusqu'à ce que la musique s'arrête.

 Regardez cette [vidéo](#) pour en savoir plus sur les boucles while.

Mise en pratique (40 minutes)

Maintenant, utilisez Swift Playgrounds : Apprendre à coder 1 pour résoudre les puzzles cochés en vert dans la liste de droite.

Pensez-y : quand avez-vous utilisé des boucles for et des boucles while ? Comment avez-vous choisi ?

Boucles while

Introduction	✓
Exécution du code pendant...	✓
Boucles While plus intelligentes	✓
Recherche de l'outil adéquat	✓
Quatre par quatre	
Faire le tour	
Terre d'abondance	
Imbrication de boucles	✓
Rectangles aléatoires	
À droite toute !	

Boucle while : boucle qui exécute un bloc de code tant qu'une condition donnée est vraie. Lorsque la condition est fausse, la boucle s'interrompt.



Penser encore et encore : boucles while

Pour aller plus loin

4

Cache-cache, bis (1 séance)

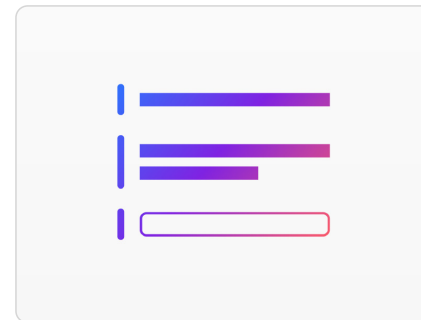
1. Cachez un petit objet dans la pièce ou à proximité.
2. Postez-vous à un endroit précis et utilisez la caméra de l'iPad pour vous enregistrer en train de donner des instructions permettant de trouver l'objet. Les instructions doivent partir de l'endroit où vous vous tenez. Utilisez des fonctions, des boucles for et des boucles while dès que vous le pouvez.
3. Maintenant, échangez votre vidéo avec celle d'un autre élève. Regardez sa vidéo pour trouver l'objet qu'il a caché. L'avez-vous trouvé ?
4. Revoyez ensemble les deux vidéos. Indiquez les endroits où vous avez utilisé des boucles for et ceux où vous avez utilisé des boucles while dans vos instructions.

Pensez-y : cette fois-ci, l'utilisation de boucles a-t-elle facilité les choses ? Ou cela les a-t-elle compliquées dans certains cas ?



21 questions (1 séance)

1. Réexaminons le Point de départ Réponses. [Téléchargez cette version du jeu.](#)



2. Ce modèle Réponses a été préparé pour vous permettre de jouer au jeu des 21 questions. Regardons d'abord le code. Que va-t-il faire ?
3. Le créateur choisit un objet ou un élément qu'il indique comme réponse dans le code. Le joueur peut poser au créateur 21 questions appelant une réponse par oui ou par non.
4. Après chaque question, le joueur peut faire une proposition dans la vue en direct pour vérifier sa réponse. Veillez à ce que le créateur affiche son playground en mode plein écran dans la vue en direct de sorte que le joueur ne puisse voir ni le code ni la bonne réponse.

Pensez-y : quels concepts de programmation ont été utilisés dans ce playground, et de quelle façon l'ont-ils été ?

Penser une même idée : algorithmes




5

Introduction (5 minutes)

En groupe, nommez certaines choses que vous faites tout le temps et qui nécessitent plusieurs étapes : par exemple, se brosser les dents ou préparer un sandwich. Il s'agit dans tous les cas d'algorithmes.

Choisissez un exemple et demandez à plusieurs élèves de donner les instructions pour accomplir la tâche. Ont-ils donné les mêmes instructions ? En quoi celles-ci étaient-elles différentes ? Ont-elles toutes abouti au même résultat ?

 Regardez cette [vidéo](#) pour en savoir plus sur les algorithmes.

Mise en pratique (40 minutes)

Maintenant, utilisez Swift Playgrounds : Apprendre à coder 1 pour résoudre les puzzles cochés en vert dans la liste de droite.

Pensez-y : selon vous, combien y a-t-il de façons de résoudre chaque puzzle ? Qui a l'algorithme le plus court ? Qui a le plus intéressant ?

Algorithme : ensemble de règles ou d'instructions à suivre pas à pas.

Pseudo-code : description informelle du code ou d'un concept, conçue pour être lue par un humain.



Algorithmes

Introduction	<input checked="" type="checkbox"/>
La règle de la main droite	<input checked="" type="checkbox"/>
Ajustement de ton algorithme	<input checked="" type="checkbox"/>
Conquête d'un labyrinthe	<input checked="" type="checkbox"/>
Où tourner ?	
Roule à droite, roule à gauche	

Penser une même idée : algorithmes

Pour aller plus loin

5

Qui est le plus grand de la classe ? (1 séance)

1. Répartissez les participants en petits groupes. Chaque groupe devra trouver un moyen, ou un algorithme, permettant de déterminer qui est la personne la plus grande de la classe. Si vous le savez déjà ou si ça se voit, ça ne compte pas !
2. Puisez dans vos connaissances en programmation pour proposer les étapes de votre algorithme. Vous pouvez inventer votre propre pseudo-code pour écrire votre algorithme en utilisant le plus possible de termes de programmation.
3. Demandez à chaque groupe de présenter son algorithme. Tous les groupes doivent le faire.

Pensez-y : quel algorithme semble le plus efficace ? Si vous deviez identifier l'élève le plus petit, que faudrait-il modifier dans votre algorithme ?



Parrot (1 séance)

1. Si vous disposez d'un drone Parrot, téléchargez la leçon Parrot de Swift Playgrounds.



2. Mettez en œuvre toutes les notions de programmation que vous avez apprises jusque-là pour faire faire à votre drone les choses suivantes : décoller, atterrir, aller dans toutes les directions et effectuer des figures acrobatiques.
3. Enfin, créez un algorithme déterminant la trajectoire de vol permettant à votre drone d'aller d'un point A à un point B.

Pensez-y : en quoi la vitesse du drone a-t-elle affecté vos trajectoires de vol ?

Penser comme un newsbot : variables

6



Introduction (5 minutes)

Imaginez que vous deviez déménager. Vous placeriez sans doute vos affaires dans des boîtes en carton, en les étiquetant d'un ou deux mots pour en décrire le contenu. Si, par exemple, vous mettez vos trophées dans une boîte, vous écririez sans doute « Trophées » sur cette boîte. Cet exemple illustre trois caractéristiques importantes d'un conteneur.

Lorsqu'on programme un ordinateur, on n'utilise pas des cartons, mais ce qu'on appelle des « variables ». Les variables sont semblables à des boîtes : elles ont une étiquette (un nom) et un contenu (une valeur). La valeur (ou contenu) peut changer ; pas l'étiquette (ou nom). On peut trouver la valeur d'une variable en trouvant la variable portant le bon nom.

 Regardez cette [vidéo](#) pour en savoir plus.

Mise en pratique (40 minutes)

Maintenant, utilisez Swift Playgrounds : Apprendre à coder 2 pour résoudre les puzzles cochés en vert dans la liste de droite.

Pensez-y : en quoi l'utilisation de variables vous a-t-elle aidé pour l'app ?

Variable

Introduction	<input checked="" type="checkbox"/>
Garder le compte	<input checked="" type="checkbox"/>
On fait monter les valeurs	<input type="checkbox"/>
Incrémenter de la valeur	<input checked="" type="checkbox"/>
Recherche de sept gemmes	<input checked="" type="checkbox"/>
Trois gemmes, quatre interrupteurs	<input type="checkbox"/>
Recherche de valeurs égales	<input checked="" type="checkbox"/>
Activation d'interrupteurs	<input type="checkbox"/>
Tout collecter	<input type="checkbox"/>

Variable : conteneur portant un nom et stockant une valeur. La valeur peut évoluer dans le temps.



Penser comme un newsbot : variables

Pour aller plus loin

6

Newsbot (1 séance)

1. Pour cette activité, vous allez créer un newsbot, c'est-à-dire un robot capable de rédiger automatiquement un bref article. Tout d'abord, pensez à un fait d'actualité ou à un événement sportif et aux informations dont vous auriez besoin pour écrire un article à ce sujet.
2. Réfléchissez ensemble à quatre à six variables, comme le nom de l'équipe, le score ou encore la date de l'événement. Rédigez un article de deux à trois phrases en utilisant ces variables, que le newsbot pourra ensuite compléter pour d'autres événements similaires.
3. Faites des équipes de deux. L'un de vous fournit les informations pour chaque variable permettant d'écrire l'article. Ensuite, inversez les rôles. Obtenez-vous deux articles complets ayant du sens ?

Pensez-y : certains noms de variables ont-ils mieux fonctionné que d'autres ? Pourquoi ?

```
VARIABLES:
• teamOneName
• teamTwoName
• teamOneFinalScore
• teamTwoFinalScore
• ballfieldName

STORY:
The teamOneName and the teamTwoName faced off at
ballfieldName. The final score was teamOneName
teamOneFinalScore to teamTwoName teamTwoFinalScore.
```

Sphero Arcade (1 séance)

1. Si vous disposez d'un robot Sphero SPRK+, téléchargez la leçon Sphero Arcade de Swift Playgrounds.



2. Appliquez des fonctions et des variables pour viser, détecter des collisions et, au bout du compte, élaborer votre propre version du jeu Pong.
3. Sur la page Jouer au Sphero Pong, analysez le code avant de l'exécuter ou de le modifier. Que fait chaque commande ?
4. Quels éléments du jeu avez-vous modifiés à l'aide de variables ?

Pensez-y : quels autres jeux pourriez-vous adapter comme vous l'avez fait avec Pong ?

Penser comme un architecte : types



7

Introduction (5 minutes)

Combien de types de bâtiments connaissez-vous ? Choisissez-en un. Qu'est-ce qui le rend unique ? En d'autres termes, quelles sont les caractéristiques (fonctionnalités) particulières de ce type de bâtiment ? Que se passe-t-il généralement dans ce type de bâtiment ? C'est ce que nous appellerons des comportements. Par exemple, une école a des classes (caractéristiques), et la cloche y sonne entre deux cours (comportement). Tout le monde est-il d'accord sur les propriétés et les comportements ? Pourquoi ?

Si nous utilisons un programme informatique pour nous aider à construire le bâtiment, nous devons être très précis. Nous devons en définir le type en en fournissant les propriétés (que nous avons appelées *caractéristiques*) et les méthodes (que nous avons appelées *comportements*).

 Regardez cette [vidéo](#) pour en savoir plus.

Mise en pratique (40 minutes)

Maintenant, utilisez Swift Playgrounds : Apprendre à coder 2 pour résoudre les puzzles cochés en vert dans la liste de droite.

Pensez-y : quels étaient les types représentés dans l'app ? Qu'avez-vous initialisé ?

Type : ensemble nommé de propriétés (fonctionnalités) et de méthodes (comportements) de données d'un certain genre.

Initialisation : création d'une nouvelle instance d'un type, ce qui implique la définition des valeurs initiales des propriétés du type concerné.



Types

Introduction	✓
Désactivation d'un téléporteur	✓
Téléporteur activé et désactivé	
Réglage d'un téléporteur	✓
Aux quatre coins du monde	
Apparition de gemmes	

Initialisation

Introduction	✓
Initialisation de ton expert	✓
Entraînement de ton expert	
Instances de différents types	✓
À deux, c'est mieux !	

Penser comme un architecte : types

Pour aller plus loin

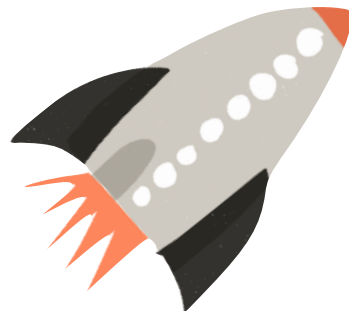
7

Devenez architecte (1 séance)

1. Choisissez un type de bâtiment, réel ou imaginaire.
2. Pensez à cinq ou six variables décrivant l'aspect de votre type de bâtiment. Ces variables ne doivent contenir aucune valeur. Elles doivent se contenter de décrire ces valeurs : comme « hauteur » ou « nombreDeFenêtres ».
3. Ajoutez des valeurs à côté des variables. Cela décrit une instance particulière du type de bâtiment. En terminologie Swift, on dit que vous initialisez une instance du type de bâtiment.
4. Terminez l'initialisation en utilisant une app de dessin telle que Notes pour dessiner le type de bâtiment, en reflétant les variables et les valeurs.
5. Trouvez-vous un partenaire et échangez les listes de variables et de valeurs de vos types de bâtiment. Dessinez chacun le type de bâtiment de l'autre, puis échangez vos dessins. Se ressemblent-ils ?

Pensez-y : voyez-vous des façons de les rendre plus ressemblants ?

Type : maisonFusée
nombreDeMoteurs = 4
hauteur = 15 mètres
nombreDeFenêtres = 8
couleur = Gris Sidéral
formePorte = rectangle arrondi



Pierre, feuille, ciseaux (1 séance)

1. Téléchargez le défi Pierre, feuille, ciseaux dans Swift Playgrounds.



2. Explorez la leçon. Comment le code fonctionne-t-il ? Quels concepts de programmation y reconnaissez-vous ?
3. Maintenant, essayez de créer une version du jeu dans laquelle vous utiliserez vos propres règles et à laquelle vous ajouterez de nouvelles actions. Pour ce faire, vous devrez définir le type de jeu et l'initialiser parallèlement aux propriétés et méthodes associées.

Pensez-y : quelles étaient les propriétés et méthodes du jeu d'origine ? Quelles nouvelles propriétés et méthodes avez-vous ajoutées pour personnaliser le jeu ?

