# Flappy Cat

## Time Separation

1. Intro: 15 minutes
   1. But et description de l’atelier
   2. Bases de scratch
   * L’environnement Scratch est divisé en 3 sections. La section du milieu est ou se trouve les blocs qui sont le code qu’on va utiliser
   * Le principe est comme des legos, les blocs se connectent l’un à l’autre. S’il n’y a pas de trou en haut, c’est un bloc de départ et s’il n’y a pas de languette en bas, c’est un bloc de fin.
   * Pour ajouter un bloc à notre code, il faut le glisser et le laisser tomber dans notre environnement
   1. Bases de programmation (while loop, for loop, if/else:si/alors)
2. Programmer: 45 minutes
3. Conclusion, feuilles de commentaires et ranger les ordis: 15 minutes

# Introduction (Prototype)

*Bonjour tout le monde! Merci de nous avoir accueillis aujourd'hui! Mon nom est \_\_\_, c'est \_\_\_ et ceci est\_\_\_\_. Aujourd'hui, nous allons passer en revue les bases du codage en utilisant un programme appelé Scratch et à la fin de l'atelier, vous pourrez faire un jeu de travail! Pas un comme \_\_ (hip, jeu vidéo populaire) \_\_, mais encore assez impressionnant pour un programmeurr la première fois!*

*Avant de plonger là-dedans, je pense qu'il serait bon de partager un peu pourquoi nous sommes ici pour donner cet atelier en premier lieu. Nous faisons tous partie d'Esprit Science, une organisation qui s'efforce de susciter l'émerveillement des élèves du secondaire dans les domaines des sciences, de la technologie, de l'ingénierie et des mathématiques. Même si je suis sûr que nous avons déjà quelques scientifiques en herbe, développeurs de jeux et médecins dans la salle, nous voulons vous montrer à quel point il peut être amusant, stimulant et significatif de travailler dans ces domaines. À cette fin, nous donnons des ateliers comme celui-ci dans les écoles de Montréal, nous prévoyons une grande conférence pour l'année prochaine qui comprendra des ateliers pratiques et des conférenciers inspirants, et nous espérons étendre au-delà de Montréal aussi en élargissant notre réseau de dirigeants bientôt.*

*Nous sommes tous ici parce que nous aimons ce que nous étudions et faisons, et nous espérons que vous aussi! Pour ne pas mettre de pression sur vous, mais c'est la première fois que nous donnons cet atelier, alors s'il vous plaît soyez gentils! Mais aussi s'il vous plaît noter tous les commentaires que vous avez pour nous - ce que vous pensez que nous pourrions faire mieux et ce que vous avez aimé!*

*Avec ça, je le remets à \_\_\_\_\_\_\_, notre expert en codage!*

# Presentation

*Il y a beaucoup de choses à savoir pour devenir un expert en programmation, par contre il y a peu de choses à savoir pour devenir quelqu’un qui sait programmer. Il y a 2 concepts généraux qui sont important à maitriser, gérer des variables, et gérer des événements. Ce sont tous des choses qui peuvent être fait avec Scratch, alors pour commencer, regardons cet environment.*

There are a lot of things that you to know in order to become a proficient coder, but there aren’t a lot of things that you need to know in order to become someone who can code. There are only 3 key concepts that you need to understand so that you can do just about anything on Scratch: MATH, VARIABLES, EVENTS(boolean). First off, here’s what scratch looks like.

[OPEN math.sb2]

Here, we have a simple event that is already setup for us, and it simply says **Whenever space key is pressed**. So that means that anything I put under is only activated when you press space.

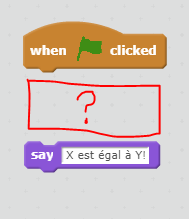
*Ici nous avons un programme qui dit, quand la touche espace est cliqué. Ceci veut dire que le code qui suit ce bloc est activé lorsque vous cliquez sur la touche espace.*

## Events

Les instructions boolean

Vous le savez déjà, mais en anglais les mots if et else veulent dire si et sinon. Avec Scratch, nous pouvons utiliser le block ‘’if \_\_\_, then \_\_\_’’ pour décider des actions que va se passer en conséquence.

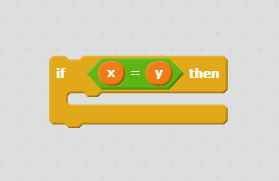
Dans le dernier exemple, nous avions assigné une certaine valeur à x et à y. Disons qu’ils valent tous les deux 100 maintenant. Comment peux-t-on faire le chat dire ‘’X est égal à Y’’, lorsque les deux valeurs sont les mêmes (comme dans notre situation)?



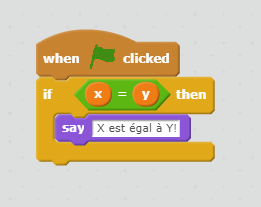
Donc nous allons d’abord utiliser l’opérateur égal, et y insérer les variables x et y.



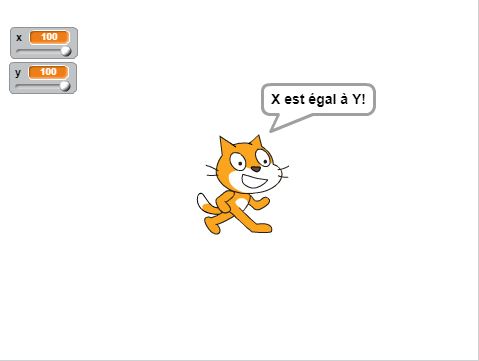
Maintenant, nous ouvrons **Control**, et allons utiliser le block **if, \_\_\_ then**, l’où on insère le block vert que l’on avait créer auparavant.



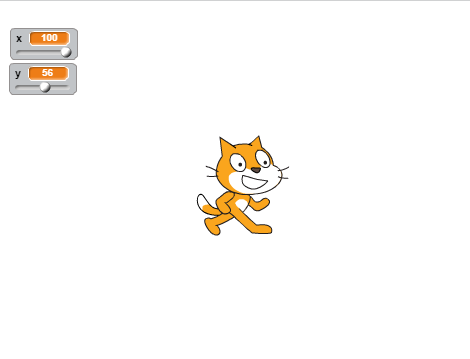
Finalement, nous ajoutons le block mauve à l’intérieur du **if, then \_\_\_**, ce qui donne



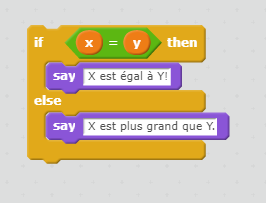
C’est l’heure de tester! Voilà ce que ca donne si on choisi 100 et 100, et l’on appuis sur le bouton vert:



Maintenant, changeons la valeur de y, et regardons ce que ca fait.



Ça marche! Mais qu’est-ce qu’on fait si on veut que le chat dit ‘’X est plus grand que Y’’ lorsque c’est le cas? Nous utilisons le bloc if/else, qui va nous permettre de performer une autre action lorsque x n’est pas égal à y. Donc l’action que nous allons performer est de dire ‘’X est plus grand que Y:

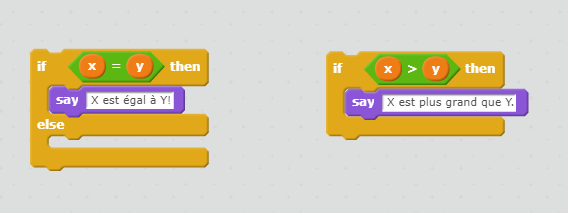


Qu’est-ce que vous pensez que ca va faire? Verifions:

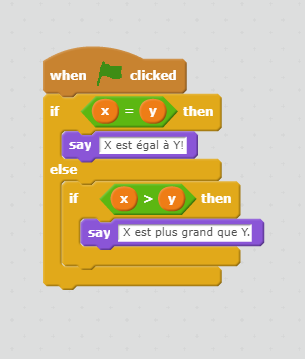


Malheureusement, ce n’est pas ce que l’on veut! En effet, nous n’avions pas vérifié si x aurait pu

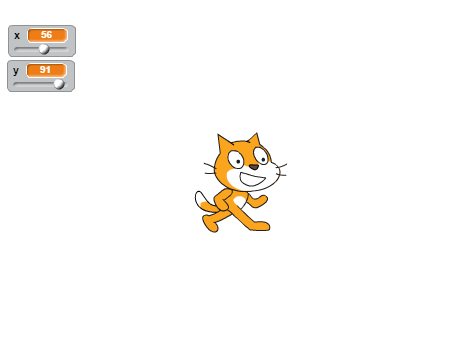
être plus petit que y; en fait, nous savons seulement si x est égal à y ou pas. Donc faisons-ça:



Le bloc à droite nous permet de vérifier si X est plus grand que y, et si c’est le cas, ça va le dire. Maintenant, il ne nous reste plus qu’à placer le bloc de droite sous ‘’else’’ du bloc gauche:



C’est le temps de vérifier:



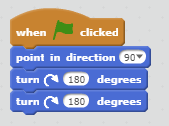
Cette fois, ça marché!

**Les boucles for/while**

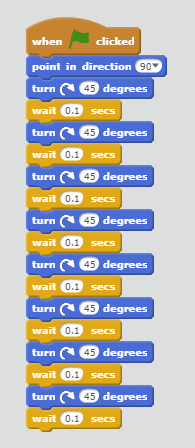
Un boucle ‘’for’’ est utilisé lorsque nous nous avons besoin de faire fonctionner un morceau de code exactement un certain nombre de fois. Prenons par exemple le chat. On veut le faire tourner sur soi-même une fois. Voici le code que l’on pourrait essayer:



Mais lorsque l’on teste le chat, rien n’arrive! C’est parce que nous avons seulement changer sa direction de 360 degrés, donc ça va retourner à sa place initiale. Essayer, à la place, de le faire tourner de 180 degrés deux fois:



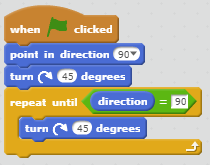
C’est déjà mieux. Et si l’on le fait tourner 4 fois de 90 degrés? 8 fois de 45 degré? Dans le fond, ça va être de mieux en mieux. Mais voici ce que le code va ressembler:



Comment pourrions nous rendre ça plus ‘’beau’’? En utilisant des boucles for! Dans scratch, tu sélectionnes Control, et le block **repeat \_\_**, et tu choisis le nombre de fois que tu veux répéter le code. Disons que l’on veut le faire répéter 8 fois, et à chaque fois on fait tourner le chat de 45 degré. Voici ce que ça ressemble:



C’est beaucoup mieux! Et si, au lieu de le faire tourner le chat exactement 8 fois, si on le fait tourner jusqu’à ce qu’il se rend à sa place initiale? Comment est-ce qu’on le pourrait faire ça? Essayons ça:



Ça marche! Dans le fond, un boucle while est utilisé lorsque nous avons besoin de faire fonctionner un morceau de code plusieurs fois, et qui s’arrête lorsque la condition que nous l’avons donné est réalisé. Dans notre cas, le boucle while est représenté par le bloc repeat until \_\_\_, et le condition est que le chat s’arrête de tourner lorsqu’il fait face à la droite.

# Conclusion

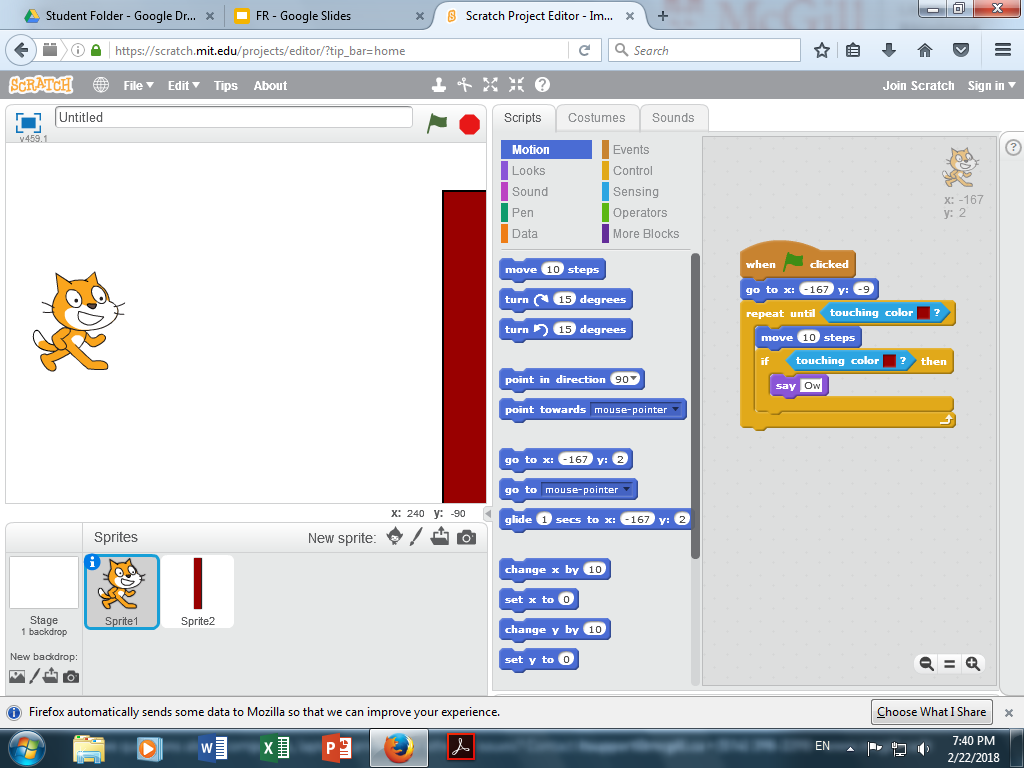
Juste avant de vous laisser partir pour la période Élite, nous avons quelques petits commentaires à faire. Tout d’abord, merci d’avoir participer à notre atelier, nous espérons que vous avez aimé.

Ce serait très apprécié d’avoir vos commentaires, qu’ils soient bons ou mauvais. Nous tentons toujours de s’améliorer et votre avis est super important. Vous pourrez les écrire sur cette feuille ou le dire à votre enseignant qui pourrait nous relayer le message.

Aussi, si cette activité vous a plu sachez que le monde de la programmation est immense et en essor. Les possibilités d’emplois sont énorme! J’étudie moi-même en génie informatique, qui est le mélange entre l’électronique et la programmation, et les gars sont en informatique pure. Par contre, il est important de savoir que si vous voulez devenir un programmeur de jeu vidéo pour Ubisoft ou autre compagnie, ou travaillé en informatique, vous devez suivre le parcours de sciences et mathématiques au secondaire. Vous devez donc réfléchir à tout cela lorsque vient le temps de faire vos choix de cours l’année prochaine.

Finalement, si jamais vous avez d’autres questions ou vous voulez plus d’information, n’hésitez pas de venir nous voir ou de visiter notre site web ou notre page Facebook.

# Presentation (Changes)



* Change the (if...else) and (while) example to an animation of our character running into a wall.
* That will free us enough time so that we can accompany them through the download/upload process.
* Give them the option to either follow us in the first couple steps of coding, or work at their pace.
* To do:
  + Add sprite customization to the end, of the presentation, since too many students waste time there.
  + Add couple of challenges for more advanced students.
  + Fix errors in the step-by-step (english, stop).
  + Start thinking about migrating content to website