Enriched Module on Coding Education for Upper Primary Level

Technology Education Section
Curriculum Support Division
Education Bureau
June 2023

Primary Four 4

Teacher Version

Booklet 2

Preface

The Education Bureau actively promotes innovation and technology (I&T) education for all students. Continuous incorporation of I&T learning elements into both the primary and secondary curricula helps strengthen the cultivation of students' interest in and capability of learning information technology and I&T from an early age, equip students with 21st century skills, and unleash their creativity and potential.

To enhance I&T education, the Education Bureau has launched the "Enriched Module on Coding Education for Upper Primary Level" for schools to adopt. Designed in accordance with the revised "Computational Thinking - Coding Education: Supplement to the Primary Curriculum" published in 2020, the curriculum module helps teachers integrate I&T elements into classroom learning more systematically. Schools should conduct appropriate curriculum planning with reference to the content of the "Enriched Module on Coding Education for Upper Primary Level", and incorporate 10 to 14 hours of enriched coding education for all upper primary students every year in order to further develop their computational thinking and strengthen their I&T learning.

The "Enriched Module on Coding Education for Upper Primary Level" is adapted from learning and teaching resources of the "CoolThink@JC" project initiated and funded by The Hong Kong Jockey Club Charities Trust and co-created by The Education University of Hong Kong, Massachusetts Institute of Technology, and City University of Hong Kong. The Education Bureau is grateful for the collaboration with The Hong Kong Jockey Club Charities Trust in consolidating and drawing on the experience accumulated by the schools in the project to develop the "Enriched Module on Coding Education for Upper Primary Level" for adoption by all publicly-funded schools in Hong Kong. The Technology Education Section, Curriculum Support Division of the Education Bureau and Department of Mathematics and Information Technology of The Education University of Hong Kong co-developed the curriculum module based on the deliverables produced and experience gained in the project. Views on the content of the curriculum module were collected from the Committee on Technology Education of Curriculum Development Council and their support was sought.

The "Enriched Module on Coding Education for Upper Primary Level" covers basic coding and computational thinking concepts, namely abstraction, algorithm and automation, as well as connection with physical objects, the use of sensors and actuators to interact with the environment, etc., allowing students to develop their computational thinking as well as interest in and ability to learn I&T through the learning of coding.

This Primary 4 curriculum module, the first of three to be developed for upper primary levels (Primary 5 and 6 forthcoming), focuses on establishing a solid foundation for students' in the above basic concepts of coding and computational thinking; through coding activities, logical thinking and problem solving skills are developed, and computational thinking is cultivated. There are a total of 8 units in the curriculum module, including 6 core units, and 2 optional extension units for schools to provide opportunities for students with a higher ability or strong interest in coding to enrich their learning and deepen their understanding of coding and innovative technology. The curriculum module also includes a project-based component that allows students to apply their computational thinking and creativity, and make good use of programming and innovative technology in different contexts, thereby formulating solutions to everyday problems for the benefit of society.

The recommended lesson time of the curriculum module (excluding the extension units) for each upper primary year level is 14 hours. Please refer to Table 1 and the Appendix for the arrangement of this Primary 4 curriculum module, the recommended lesson time, as well as the pedagogy to be adopted.

Table 1: Arrangement of the Primary 4 curriculum module and recommended lesson time

		Core Unit		**Extension Unit**	
Unit	**Unit Title**	**Recommended Lesson Time (in minutes)**	**No. of Lessons (35 minutes for each lesson)**	**Recommended Lesson Time (in minutes)**	**No. of Lessons (35 minutes for each lesson)**
1	Introducing Scratch Programming	70	2		
2	Exploring Under the Sea	70	2		
3	Storytelling	70	2		
4	Space Traveling	105	3		
5	Creating a Maze Game	140	4		
6	Creating a Maze Game with micro:bit			70	2
7	Drawing Shapes in Scratch	105	3		
8	Designing Line Pattern Art			70	2
	Final Project	280	8		
		840 (14 hours)	24	140	4

Views and suggestions on the "Enriched Module on Coding Education for Upper Primary Level" are always welcome. These may be sent to:

Chief Curriculum Development Officer (Technology Education)
Curriculum Support Division
Education Bureau
Room W101, 1/F, West Block
Kowloon Tong Education Services Centre
19 Suffolk Road, Kowloon Tong
Kowloon, Hong Kong

Fax: 2768 8664
E-mail: teched@edb.gov.hk

Pedagogy

Teachers may make reference to the seven-step guide introduced in the Technological Pedagogical Content Knowledge (TPACK) framework for the teaching of computational thinking (CT). Technological content knowledge (TCK) refers to the knowledge of using block-based programming environments for coding. Content knowledge (CK) refers to the knowledge of CT concepts, practices, and attitudes to be taught. Pedagogical content knowledge (PCK) refers to pedagogies that do not involve the use of programming environments for teaching CK. TPACK refers to the integration of the use of technology and pedagogy to teach CK in context.

Based on the four dimensions of the TPACK framework above, teachers may adopt the seven-step guide in the instruction of each unit with a view to developing students' problem solving skills and digital creativity. The last three steps emphasise applying TCK to exploring the possible use of tools in the programming environments for the cultivation of digital creativity; revisiting and reviewing CK for consolidation; and reflection on PCK to engage in the improvement of teaching practices relevant to CK (Kong, Lai & Sun,2020; Kong & Lai, 2022; Kong, Lai & Li, 2023).

Step 1: TCK (Introducing features of the programming environment in a specific context)

Step 2: CK (Introducing computational thinking concepts, practices and attitudes to be taught)

Step 3: PCK (Adopting pedagogy such as allowing pre-coding access to games or apps to pave the way for reflection on the design of games or apps; and engaging in unplugged activities to enhance understanding of more difficult coding-related concepts, practices and attitudes)

Step 4: TPACK (Applying knowledge of using programming environments for teaching CK with appropriate pedagogy in a specific context)

Step 5: TCK (Encouraging students to suggest applications of relevant features of the programming environment in other contexts, thereby inspiring their digital creativity)

Step 6: CK (Helping students reflect on CT concepts, practices and attitudes to consolidate their learning)

Step 7: PCK (Conducting self-reflection on the pedagogy adopted in the unit with a view to improve the next round of teaching)

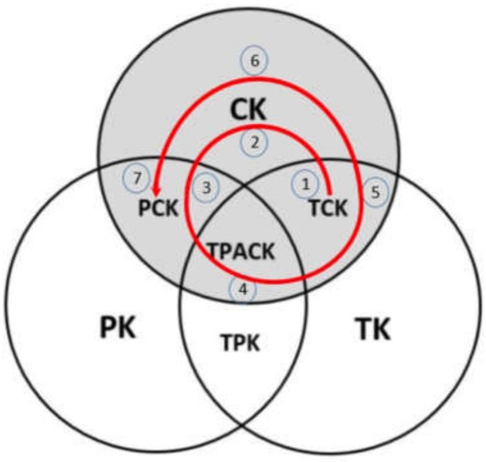


Figure 1 The seven steps in the shaded areas (CK, TCK, PCK, and TPACK) indicate those steps needed for teachers to teach content knowledge of CT. (Kong, Lai & Sun, 2020)

References

Education Bureau. (2020). *Computational Thinking - Coding Education: Supplement to the Primary Curriculum*. Hong Kong: Author.

Kong, S. C., & Lai, M. (2022). A proposed computational thinking teacher development framework for K-12 guided by the TPACK model. *Journal of Computers in Education, 9(3)*, 379-402.

Kong, S. C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education, 151*, 103872.

Kong, S. C., Lai, M., & Li, Y.G. (2023). Scaling up a teacher development programme for sustainable computational thinking education: TPACK surveys, concept tests and primary school visits. *Computers & Education, 194*, 104707.

Enriched Module on Coding Education for Upper Primary Level (Primary 4)

Editorial Board Members

Content for Booklet 2

Unit 5: Creating a Maze Game Teacher Guide

Content

Unit 5: Creating a Maze Game
Teaching Plan

Prior Knowledge

Students need to understand how to use and run Scratch from previous units. They should also know movement of sprites and changes of costume, and have some understanding of conditionals.

Learning Objectives

1. Control Scratch sprite movements using keyboard events;
2. Reuse and remix codes for costume changes and sprite coordinate movements;
3. Demonstrate an understanding of if-then conditionals by using them in a project;
4. Use "forever" block appropriately in a Scratch project to demonstration the understanding of "Iteration";
5. Utilize collision events correctly in a Scratch project;
6. Demonstrate an understanding of the use of variables to store information in a project;
7. Demonstrate an understanding that coding can be a fun and social experience through sharing their maze games with their classmates and teachers.

Learning Elements

Computational Thinking Concepts and Practices:

Key Learning Elements	Items
Abstraction	Express the algorithm
Algorithm	Problem Solving Procedures: Problem identification, Problem analysis, Algorithm design, Programming Basic Programming Constructs: Sequence, Branching / Selection, Iteration Coding Concept and Practices: Design, Reuse and Remix codes, Testing and Debugging

Coding Skills:

1. Make use of if-then statement
2. Make use of "touching" block
3. Use variable to store score.
4. Apply testing and debugging in completing each task of this unit.

Others (including Attitude):

1. Develop interest in programming;
2. Show perseverance and positivity in testing and debugging;
3. Inspire students to be creative and innovative to enhance their maze game projects.

Lesson Plan: This unit consists of 4 lessons of 35 minutes.

Teacher Preparation for the Lesson: Before the class, teachers should prepare a studio with the maze game template (with backdrop and sprites inside), so students can focus on computational thinking. Also, students can share their projects to the studio after completion.

Pre-task for Students: Before Lesson 1, teachers can assign a simple pre-lesson task on making the sprite move with keyboard presses. Alternatively, without any code blocks, teachers ask the students to move the sprite to different corners of the stage, and let the students think about how to code to do so.

Lesson 1

Time	**Activity**
5 mins	**To Play** 1. Play the Scratch project Maze Game (Demo): https://scratch.mit.edu/projects/722154863. 2. Tell students to use keyboard (arrow keys) to control the sprite and try to win the game.
10 mins	**To Think** 1. Explain the flow diagram of the maze game. 2. Ask students to complete the missing steps.
20 mins	**To Code & To Think** 1. Let students go to the studio through link and start with the maze game template (https://scratch.mit.edu/projects/727439171), so they can focus on coding but not drawing the layout. 2. Ask students to follow the instructions in the Student Guide to create a maze game. 3. Explain to students about the x and y coordinates in Scratch. 4. Review and discuss with students the pre-lesson task.

Lesson 2

Time	Activity
35 mins	**To Code & To Think** 1. Ask students to continue on the maze project they have been doing in the previous lesson. 2. In the game's current state, the sprite will simply move across the maze with no detection of collision events. 3. Build on the prior knowledge of students on Branching and Iteration, ask students to think about the conditions and steps under the two conditionals, use the new features – Touching (object) to set up the two conditionals. 4. Let students think about and explore how to constantly check for the conditions. (Add forever block to make the conditionals work fine.) 5. Teacher may also ask students to try repeat block and help students to understand "forever" block.

Teacher Preparation for the Lesson: Prepare a box and some candies to run the unplugged activity.

Lesson 3

Time	Activity
5 mins	Recap the concept and skills acquired in last lesson. Tell students what they are going to achieve in this lesson (keeping scores and finish coding the game).
15 mins	**To Learn, To Think & To Code** 1. Use "Unplugged Activity: Variables" to help students understand the concept of variables. 2. Explain to students the variables (score) concepts and tell them to add score in the game.
15 mins	Guide students to think about how the score of the game changes and let them work on the coding task. Students reflect on how to enhance the program at the end.

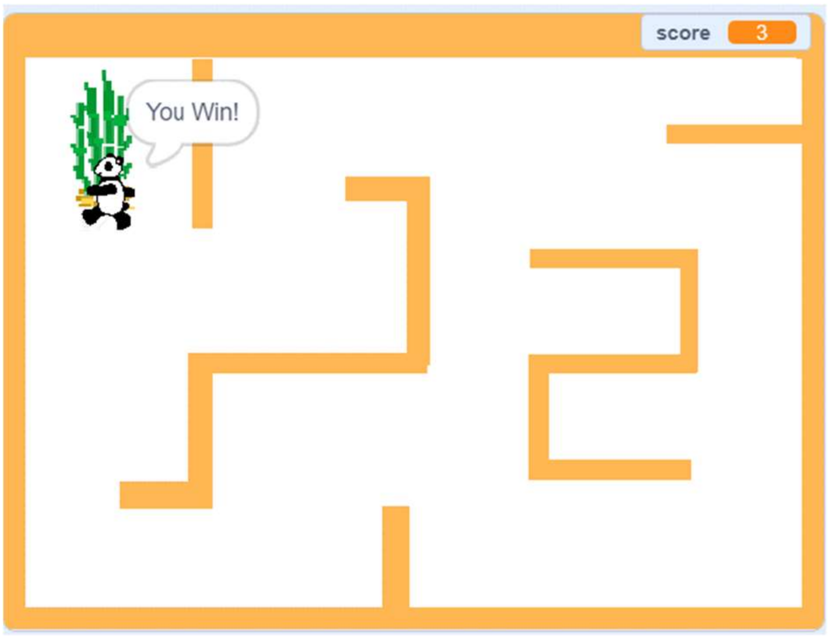
Pre-task for Students: Students think about how to design their own maze game.

Lesson 4

Time	Activity
25 mins	**To Create: Supermarket Maze** Ask students to create a new project on their own or work on another similar project: e.g. move the boy sprite to find the bread sprite in the supermarket maze.
10 mins	**To Reflect:** **Share the Projects and Provide Constructive Feedback on Program Design** 1. Create a studio and give students the studio URL. Ask students to save and submit their projects to the teacher's Studio. 2. Have one or two students share their project with the whole class. Feedback to be given by peer and teachers. 3. Students get inspired by peers' feedback and projects to enhance their maze game projects. 4. Students should share, think of ways to improve / enhance their program as well as the aesthetic values. **Review of Student Learning** 1. Review on the features of Scratch, and key concepts and practices learnt in the unit. 2. Ask students to complete the review questions, appropriate feedback should be given by teachers.

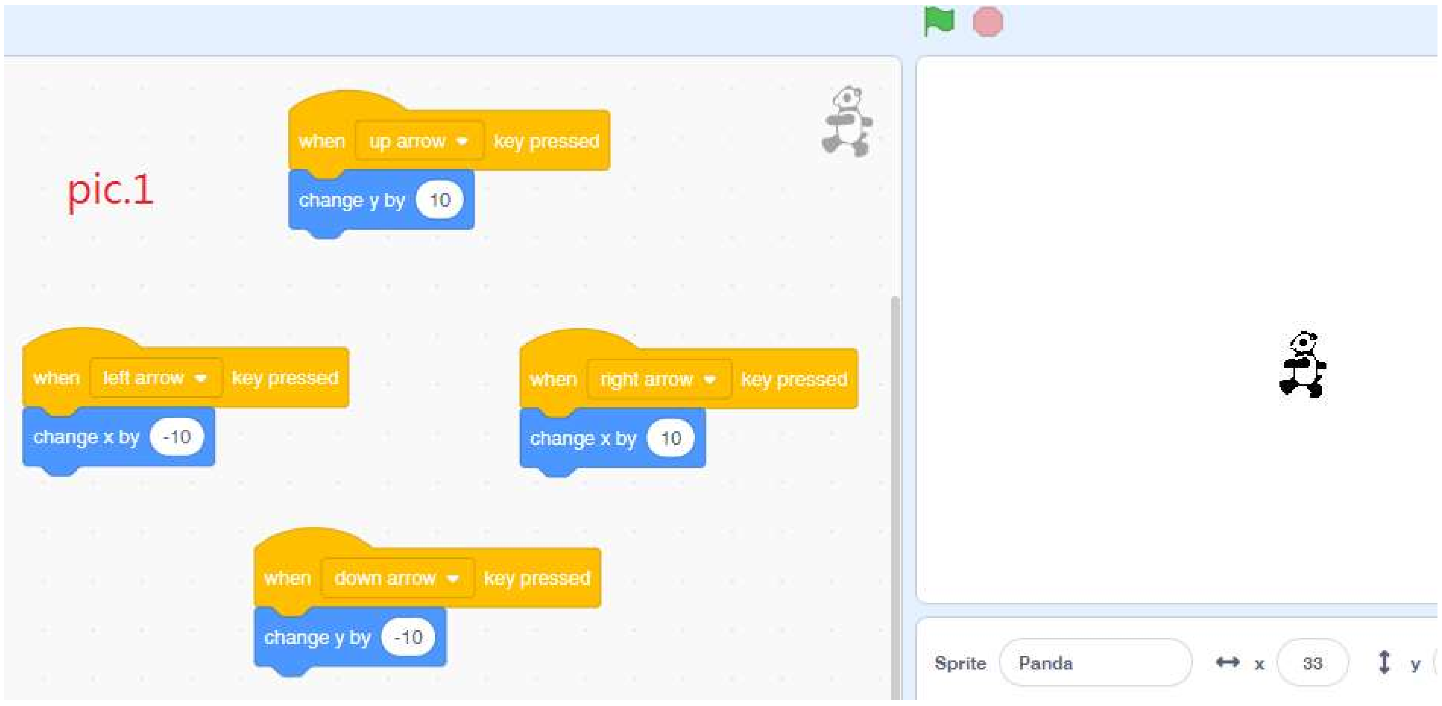
Creating a Maze Game

Let's learn to make a game with Scratch!



To Play

Suggestion of pre-lesson task:
Create a project with the blocks like pic.1, let students play it before the lesson.
Alternatively, without any code blocks, teachers ask the students to move the sprite to different corners of the stage, and let the students think about how to code to do so.


pic.1

```
when up arrow key pressed
change y by 10

when left arrow key pressed
change x by -10

when right arrow key pressed
change x by 10

when down arrow key pressed
change y by -10
```

Creating a Maze Game

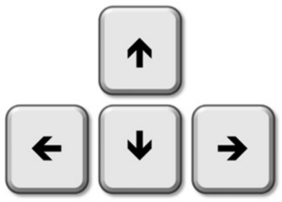
To Play

Play the Maze Game (Demo) https://scratch.mit.edu/projects/722154863
with keyboard (up / down / left / right arrow keys).

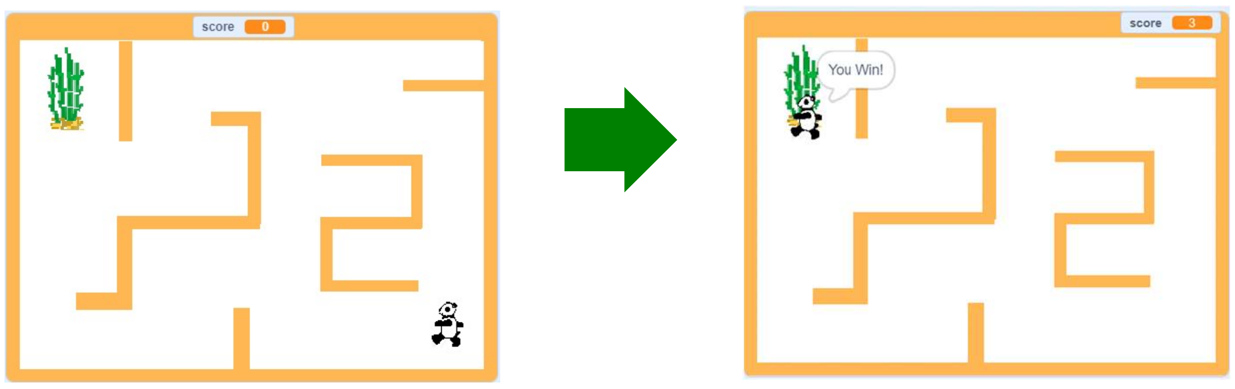
How can you win the game?

What happened when you touch the wall?

What happened when you touch the bamboo?



Teacher can give more hints if needed, e.g. the change in Score, any sounds, any message displayed, etc.

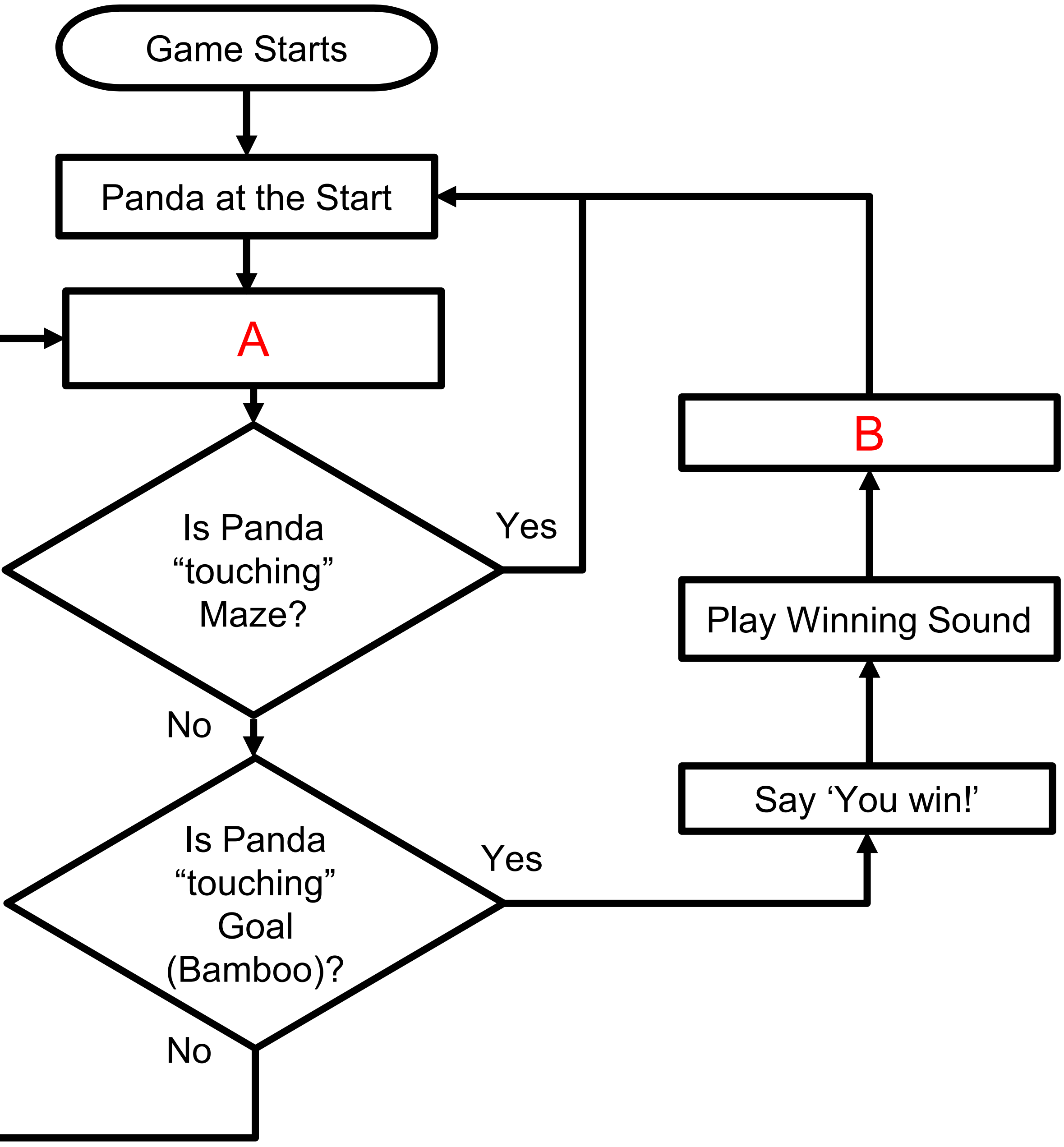


Creating a Maze Game

To Think

Complete the following flow diagram.

A. Control Panda with Keyboard	**B.** Score + 1

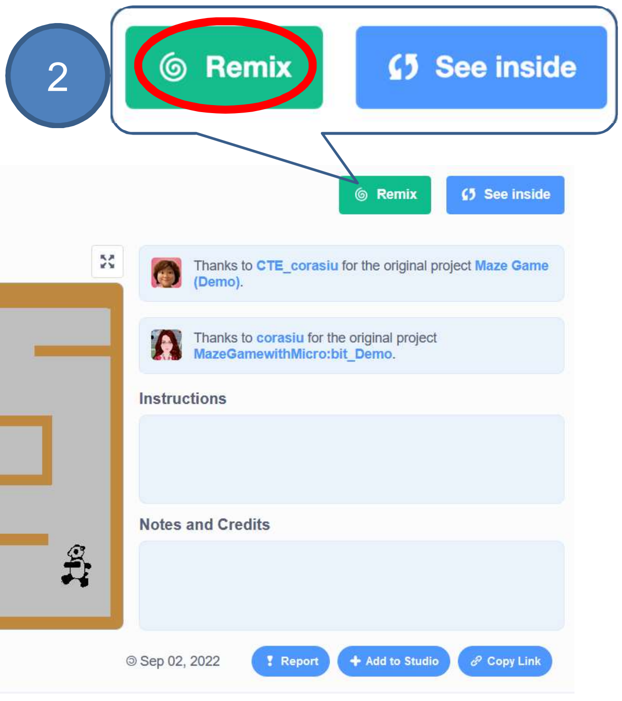


Creating a Maze Game

Start Here

1. Sign into your account at https://scratch.mit.edu/ 1

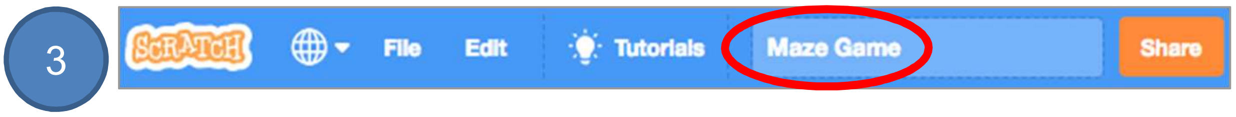
2. Open the Maze Game Template project:
https://scratch.mit.edu/projects/727439171
and click on the Remix button.



Thanks to CTE_corasiu for the original project Maze Game (Demo).

Thanks to corasiu for the original project MazeGamewithMicro:bit_Demo.

3. Rename the project as Maze Game.



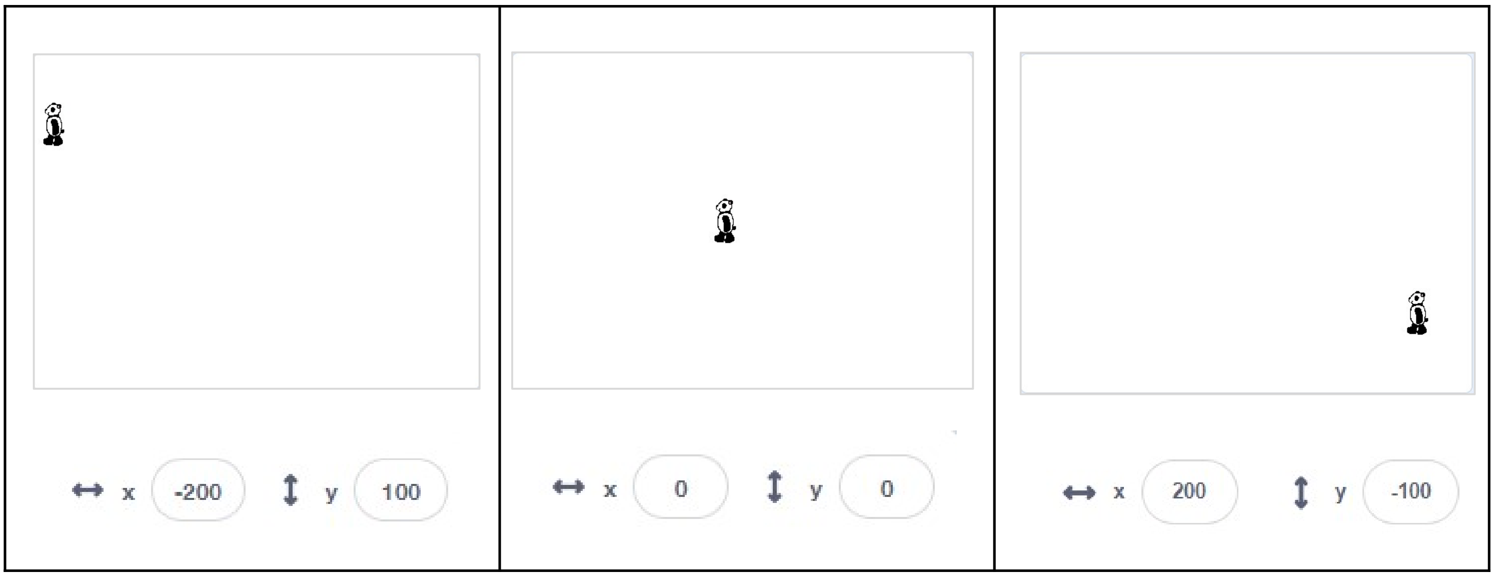
There are completed maze game map and sprites inside the template, so students can focus on coding and develop computational thinking but not drawing the layout.

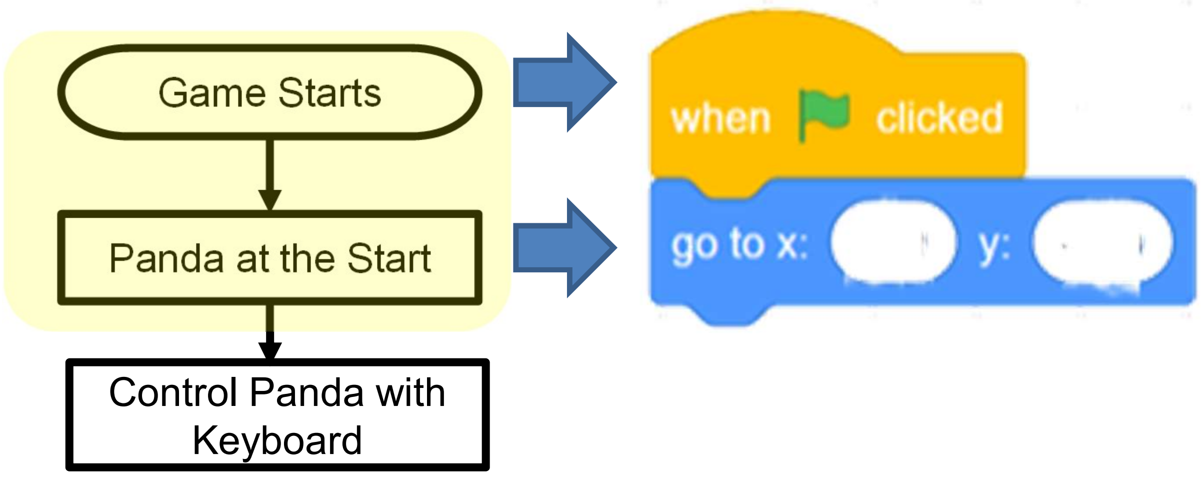
Creating a Maze Game

To Think and To Code: Panda at the Start

See Appendix P.33

Any differences between these three pandas?





Testing and Debugging

Click the green flag and see if Panda is back to the starting point.

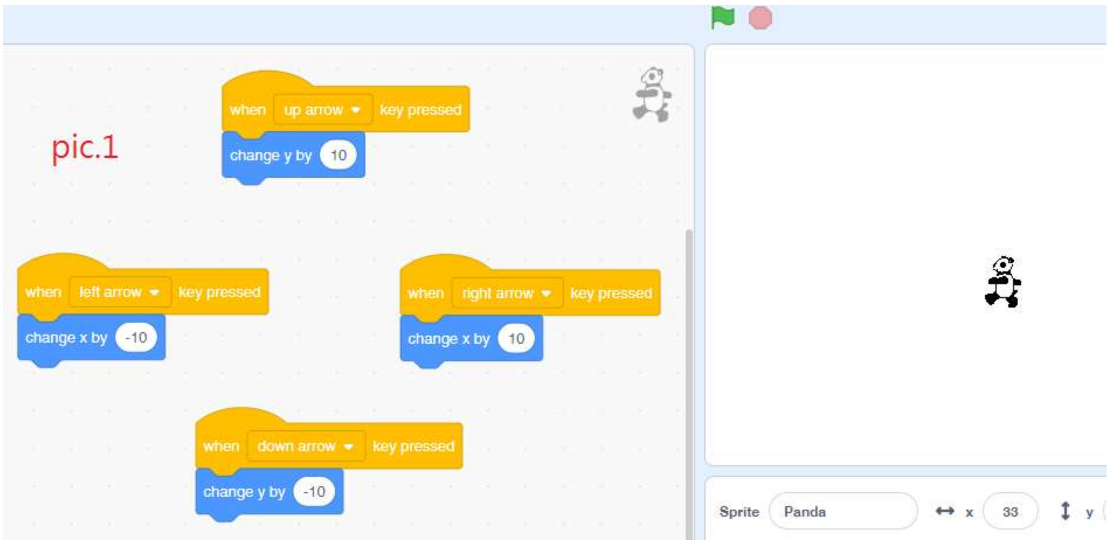
Panda at the Start

Creating a Maze Game

To Think and To Code: Control Panda with Keyboard

See Appendix P.33

Look at the blocks and backdrop:

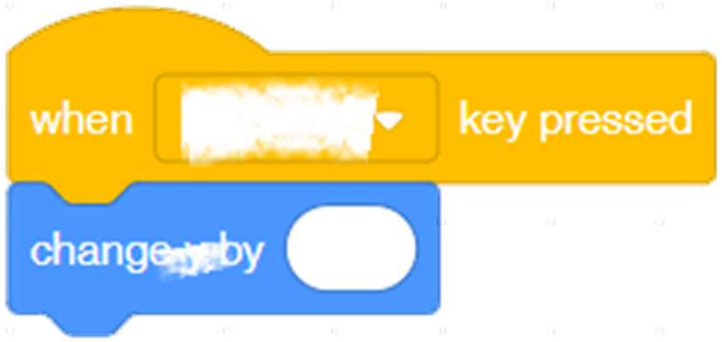

pic.1
when up arrow key pressed
change y by 10
when left arrow key pressed
change x by -10
when right arrow key pressed
change x by 10
when down arrow key pressed
change y by -10
Sprite Panda
x 33
y

Teacher can review the Pre-task and discuss with students on how to control the panda's movement. Ask them e.g.

1. Where the Panda will be when I press "up arrow" once?
2. Where the Panda will be when I press "left arrow" twice?

Student can point it out on the blackboard.

Now modify the blocks to control the Panda's movement.



Testing and Debugging

Can you control the panda with four arrow keys?

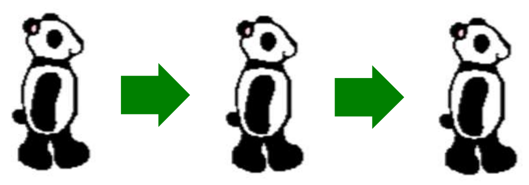
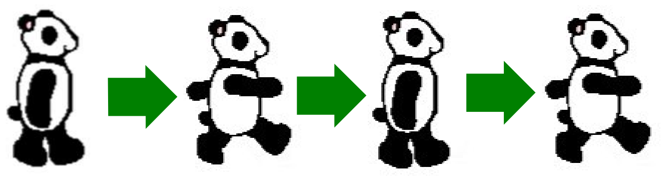
Panda at the Start → Control Panda with Keyboard → →

Creating a Maze Game

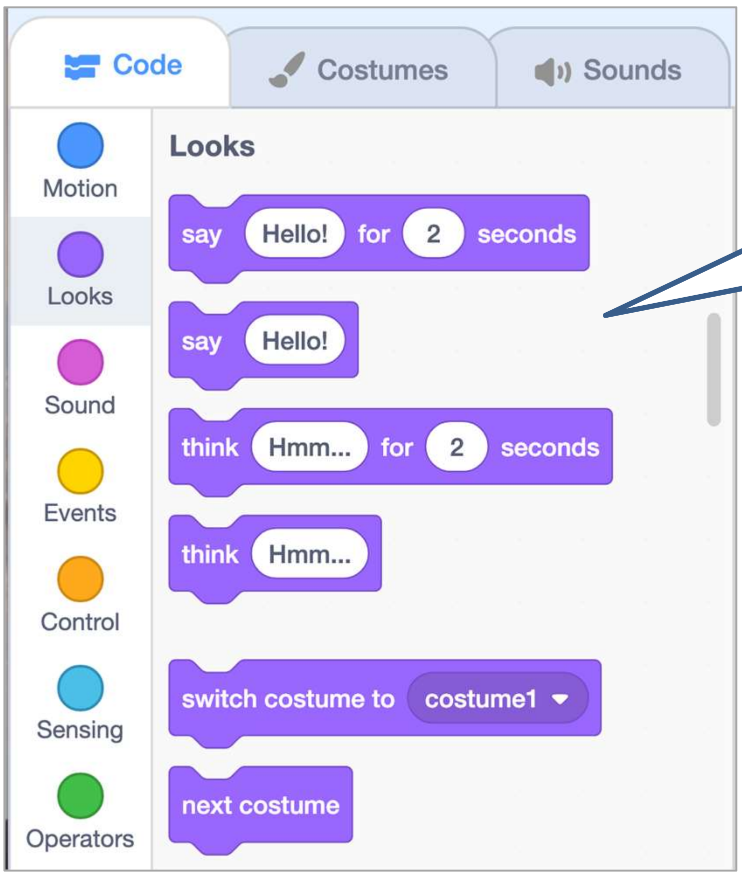
To Think and To Code: Make Panda Walk

See Appendix P.34

Which panda seems like "walking" now?

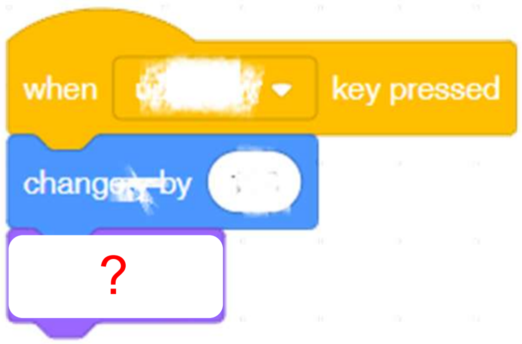
	
Panda A	Panda B

How would you change the sprite's costume to make it look like it is walking?



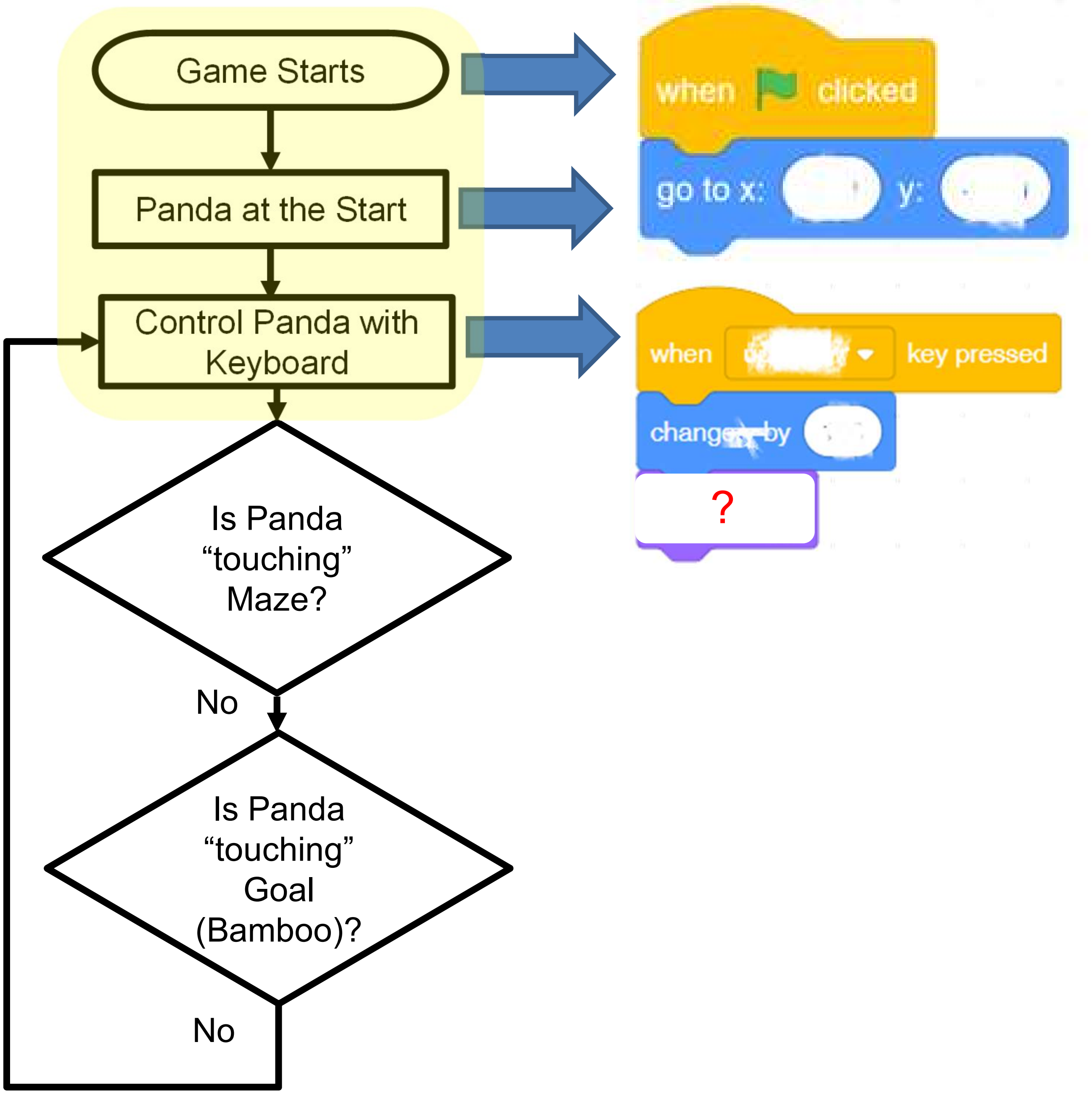
```
Looks
say Hello! for 2 seconds
say Hello!
think Hmm... for 2 seconds
think Hmm...
switch costume to costume1
next costume
```

Which block should we use to make the panda walk?

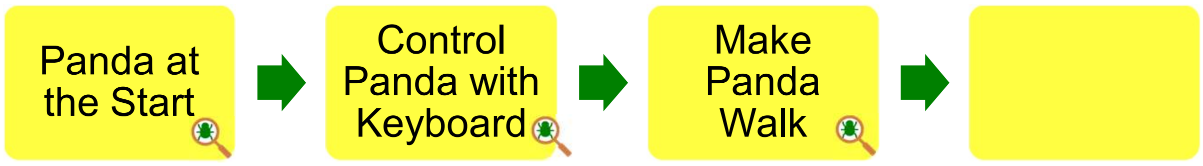


```
when [ ] key pressed
change by ( )
?
```

To Think and To Code: Make Panda Walk


Game Starts
Panda at the Start
Control Panda with Keyboard
Is Panda "touching" Maze?
No
Is Panda "touching" Goal (Bamboo)?
No
when clicked
go to x: y:
when key pressed
change by
?

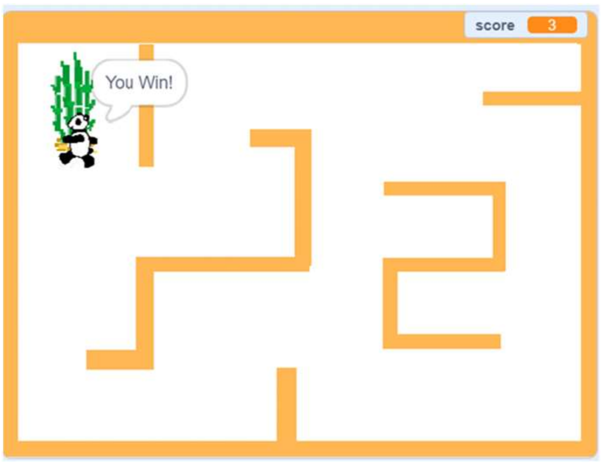
Testing and Debugging
Test to see if the Panda can walk now.



Creating a Maze Game

Let's finish our game!

In this game, you guide the panda to its bamboo forest with your keyboard, avoiding the walls of the maze. Reaching the maze earns + 1 point, hitting a wall puts panda back to the start. You will also add speech, score and sounds.



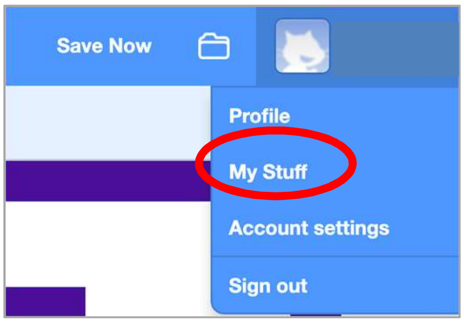
Start Here

1. Sign into your account at scratch.mit.edu.

1

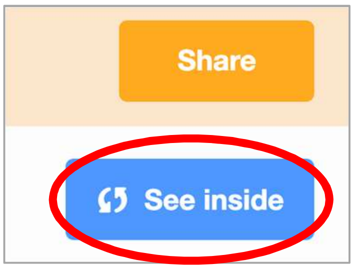
2. Go to My Stuff and open up your maze game project.

2



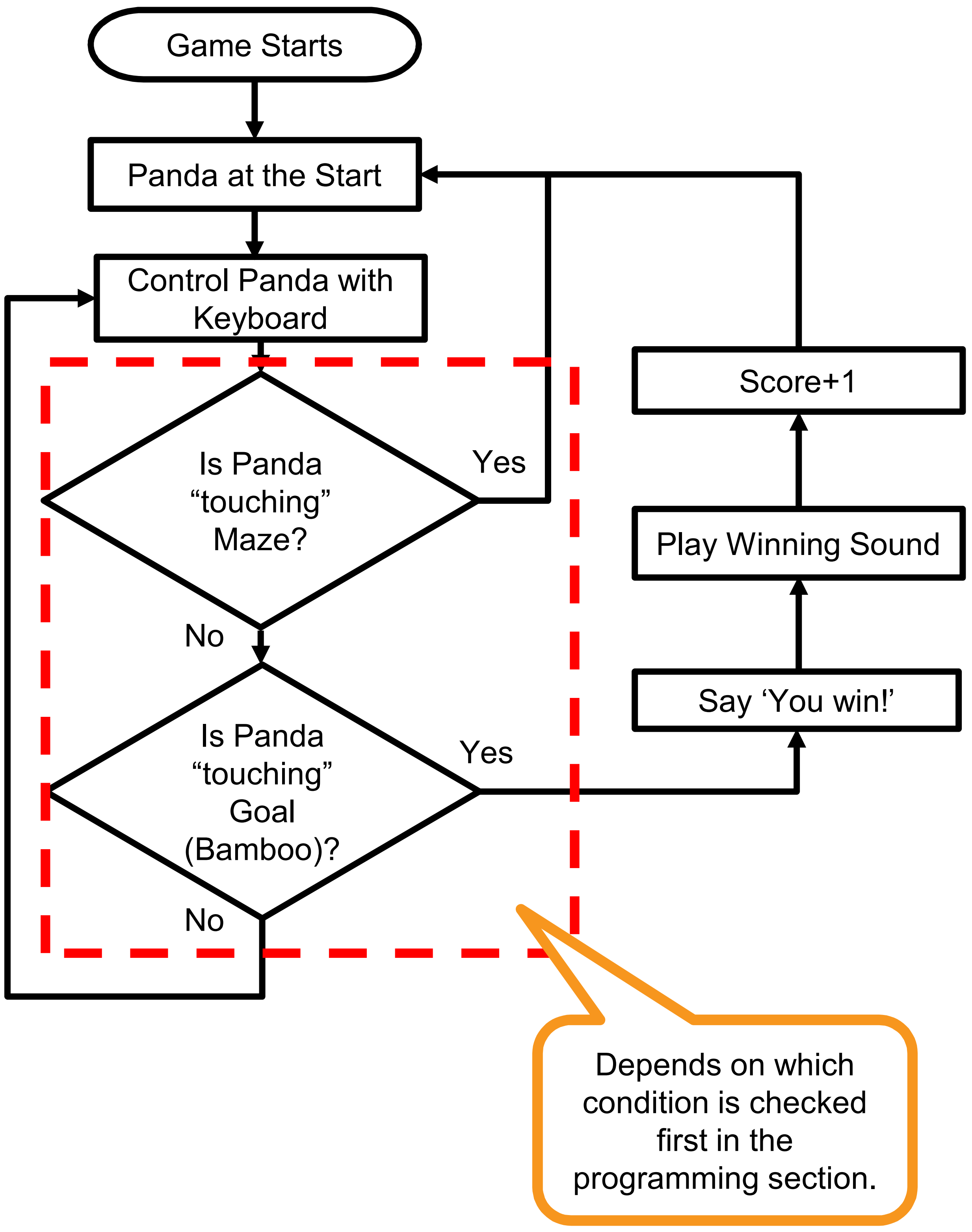
3. Click the See inside button in the top right area of the website to continue working on your game.

3



To Think: Branching / Selection (If the Panda touches…)

What will happen to the panda in the game?

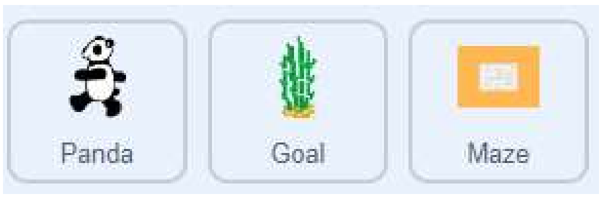

Game Starts
Panda at the Start
Control Panda with Keyboard
Is Panda "touching" Maze?
Yes
No
Is Panda "touching" Goal (Bamboo)?
Yes
No
Score+1
Play Winning Sound
Say 'You win!'
Depends on which condition is checked first in the programming section.

Creating a Maze Game

To Think: Branching / Selection (If the Panda touches…)

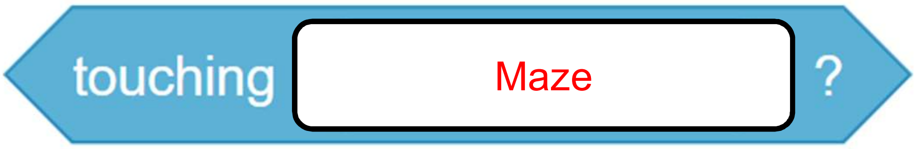
In the template, we have the three sprites:

What would they do?



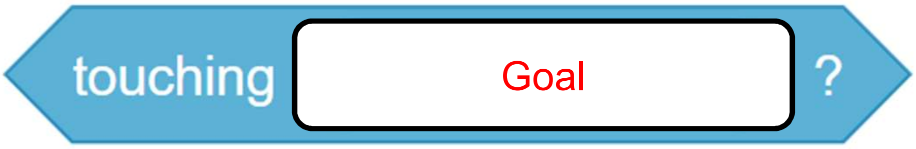
Panda is walking…

If



Then (what happens?) Back to starting point

If



Then

Panda says "You Win!", play win sound

score + 1 (or other idea)

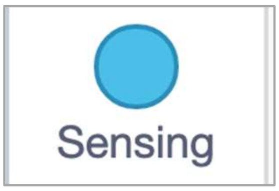
Creating a Maze Game

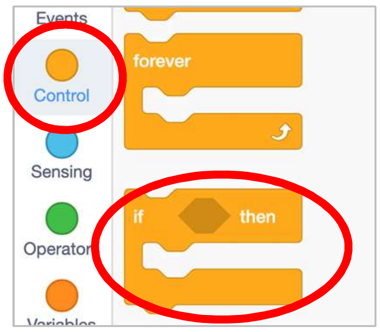
To Code: Touching Maze

See Appendix P.35

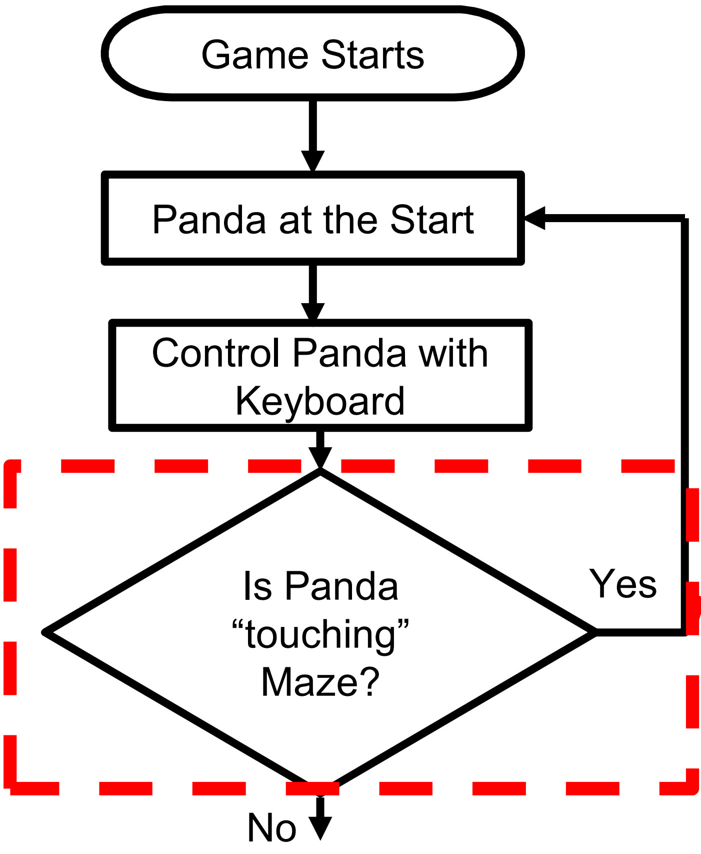
1. Throughout the game, we want the Panda sprite to act differently if it touches something. To do this, we should pull out an "if-then" block from the "Control" drawer.
2. With the algorithm you designed, what kind of conditions will trigger those actions?

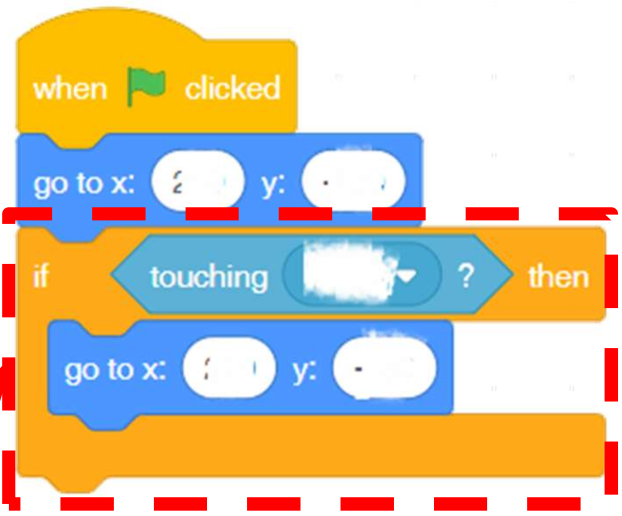
   Hint: Take a look on the "Sensing" drawer.





3. If Panda is touching Maze:



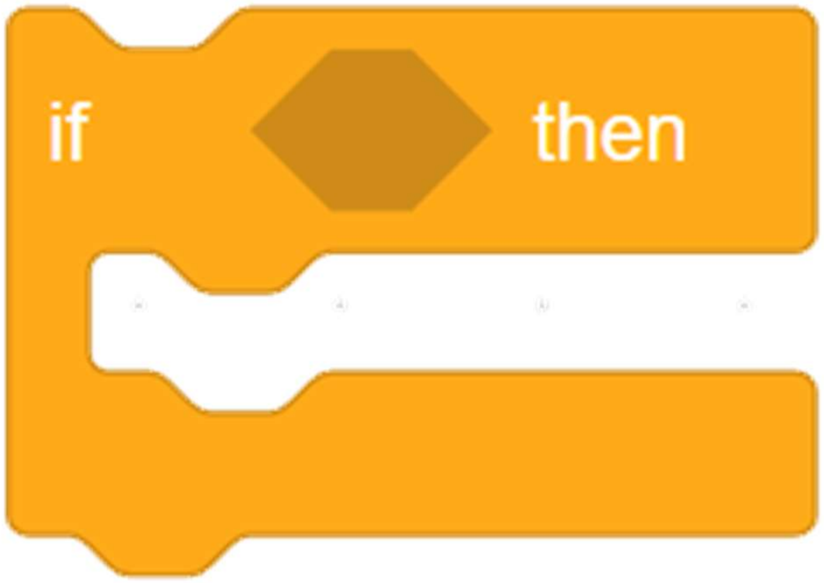


Creating a Maze Game

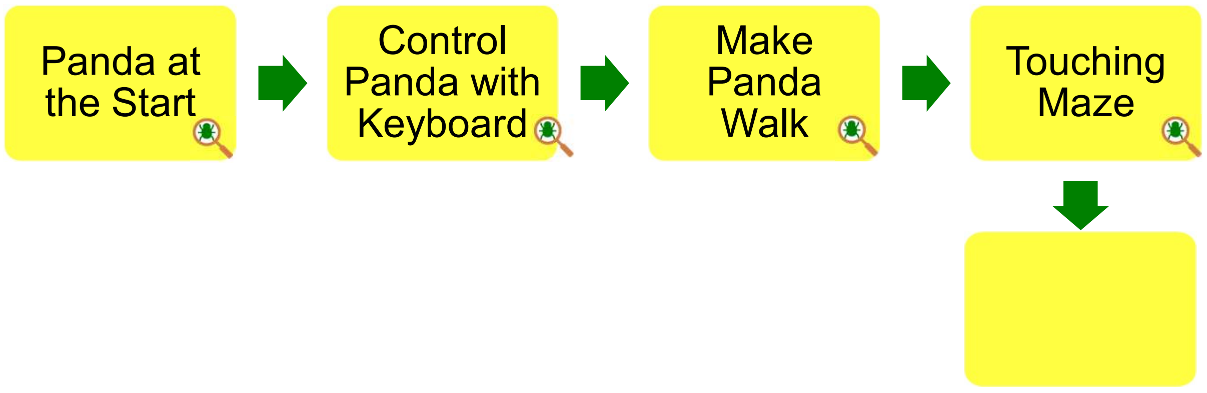
Review Branching / Selection

1. We use conditional statements in programming to enable computers to make decisions.

2. Conditionals always have an if part, which tells the program in the then part what to do when the condition is true.



Testing and Debugging
Time to test! Click the green flag and move the Panda hit the wall.
Does it go back to the starting point?
Do it again, what is difference now?

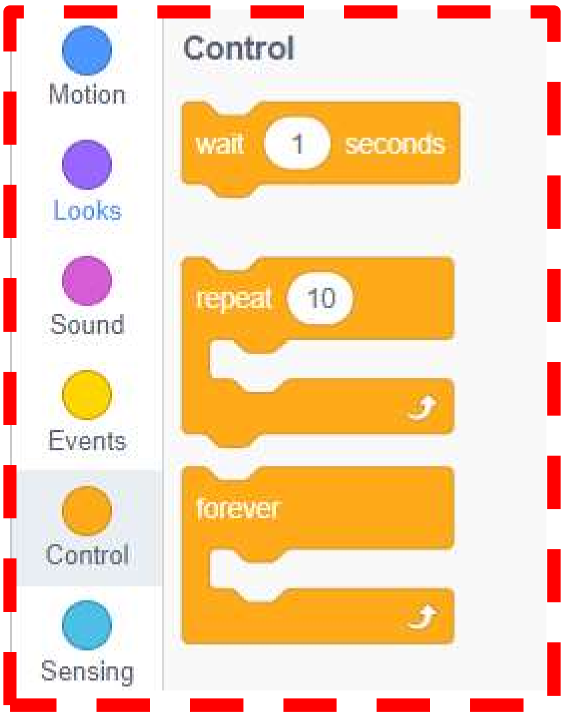


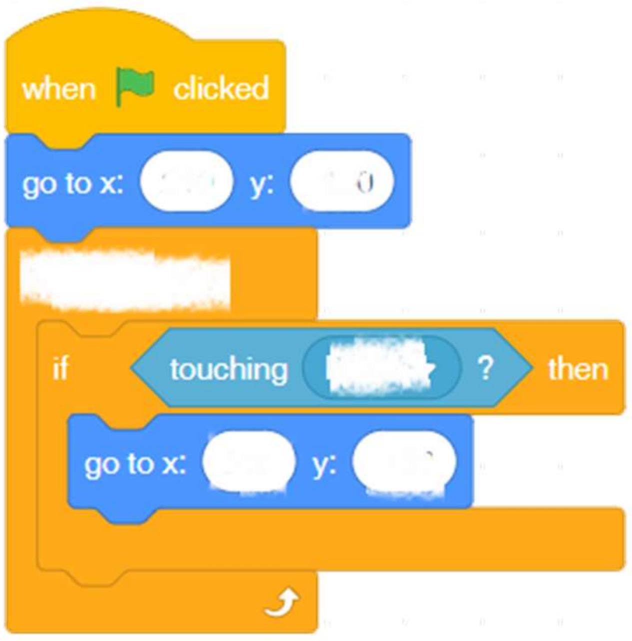
Creating a Maze Game

To Think and To Code: Iteration

See Appendix P.36

Look at the "Control" drawer, which blocks will you use to help solve this problem?





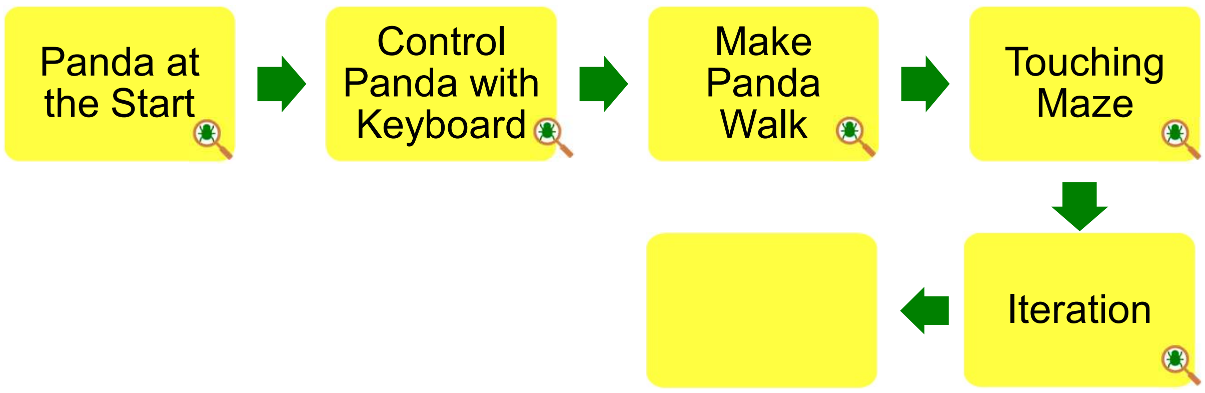
Do you need to keep checking every time the Panda walks?

☐ Yes ☐ No

Guide students think about how to keep checking the two if-then statement or let them try and understand the importance of "forever" block.

Why? Without "Forever", those other code blocks will only be executed one time, so the game will not work normally.

Testing and Debugging



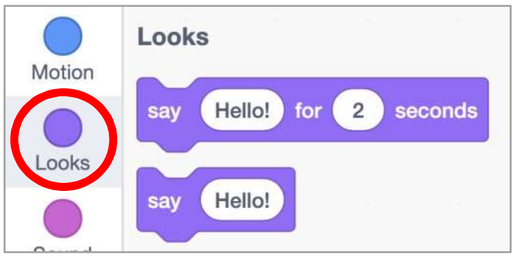
Creating a Maze Game

To Code: Touching Goal

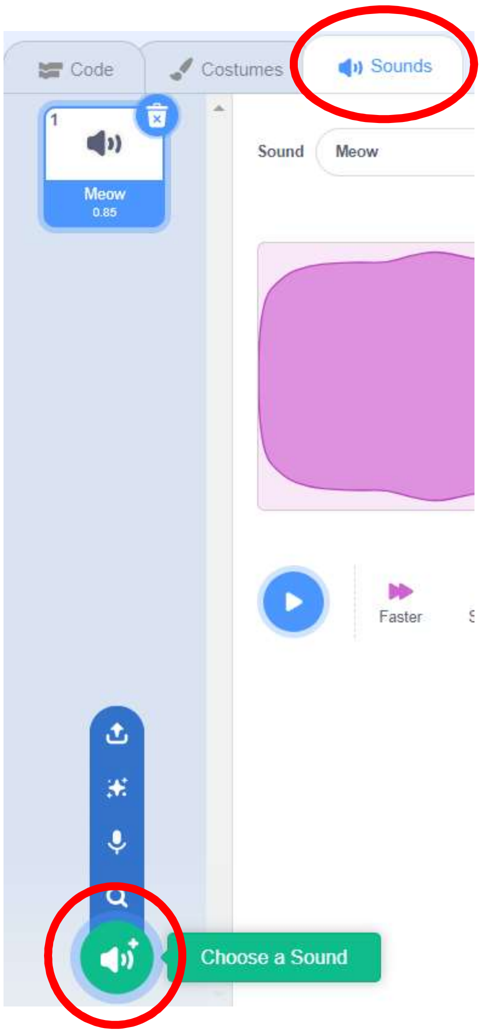
See Appendix P.37

If Panda touching the Goal (Bamboo):

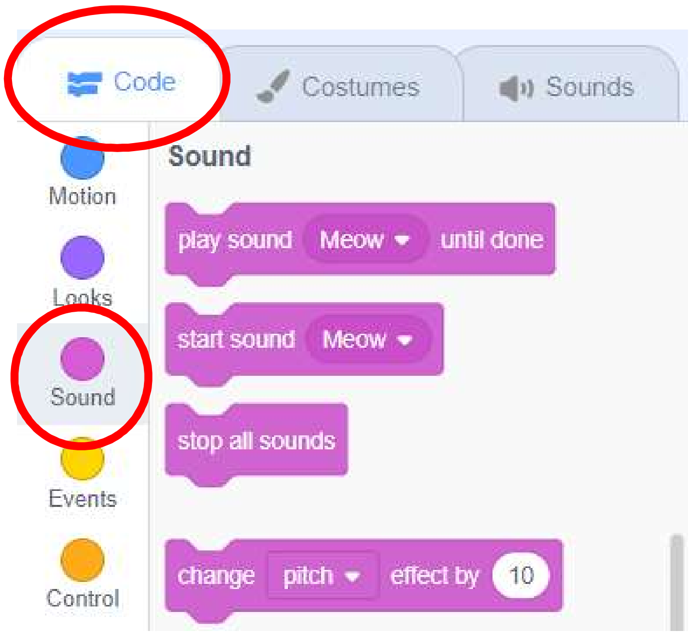
1. Make Panda say something, go to the "Looks" drawer. Which block do you use?



2. Add winning sound. Go to the "Sounds" tab. Click on the "Choose a Sound" icon at the bottom left to select a sound.



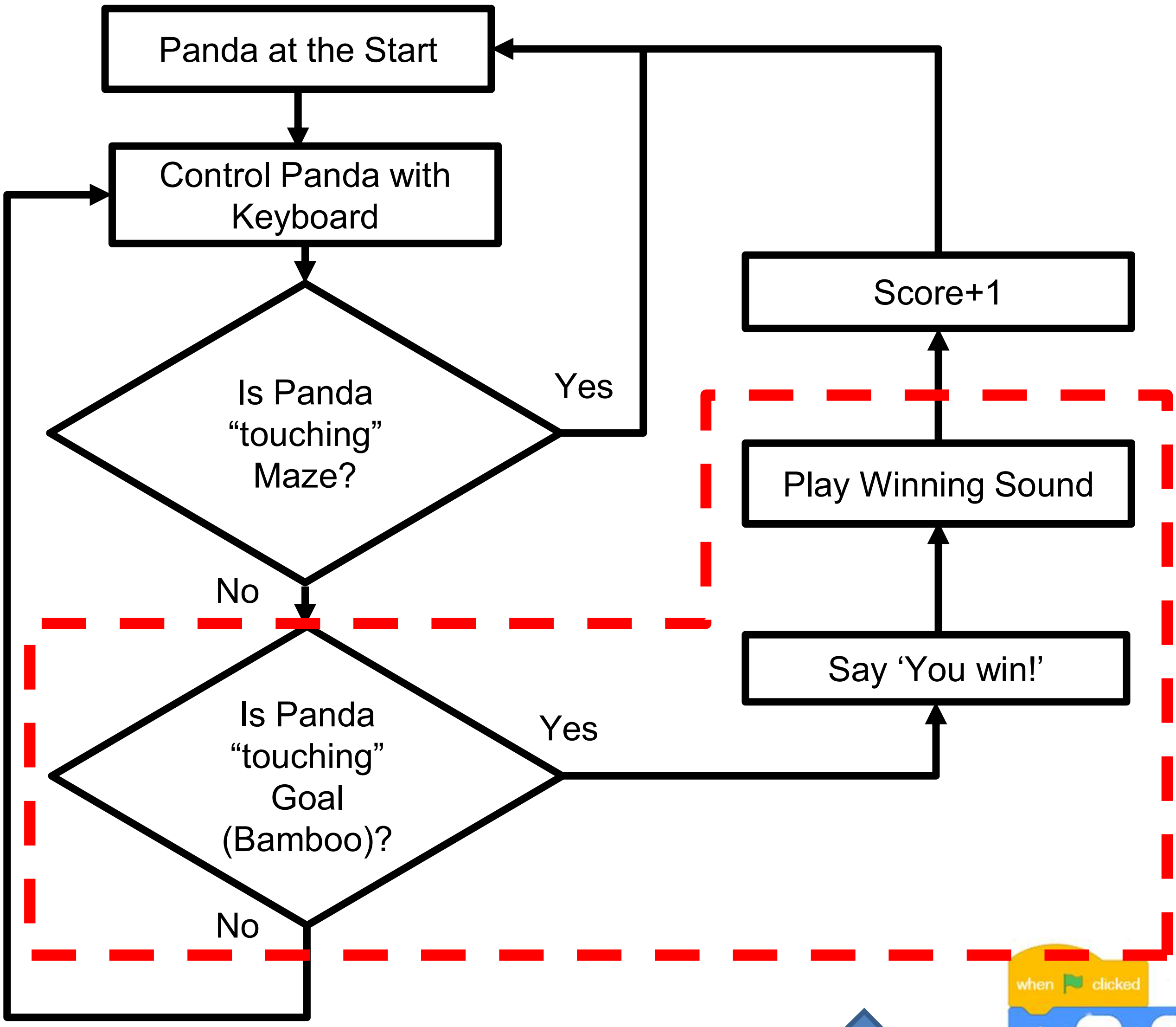
3. Do you remember how to make your sprite play the sound you found? Hint: Go to the "Sound" drawer of "Code" tab.

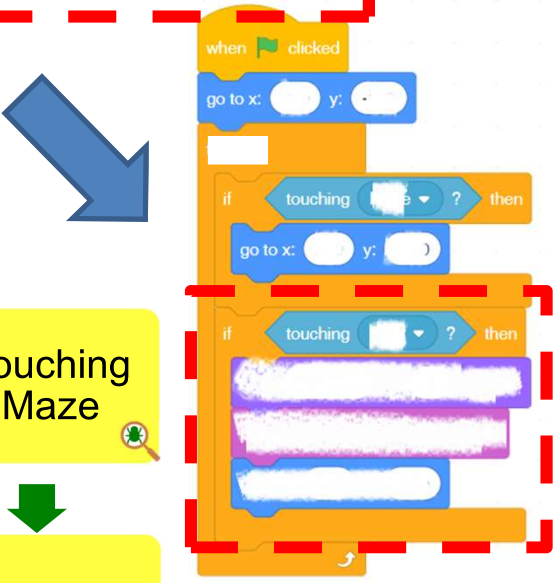


Creating a Maze Game

To Code: Touching Goal

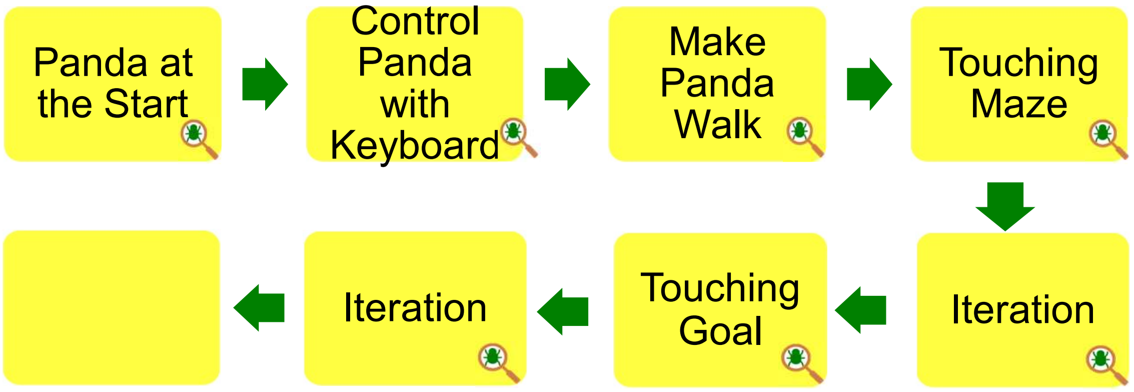
If Panda touching the Goal (Bamboo):


Panda at the Start
Control Panda with Keyboard
Is Panda "touching" Maze?
Yes
No
Score+1
Play Winning Sound
Say 'You win!'
Is Panda "touching" Goal (Bamboo)?
Yes
No



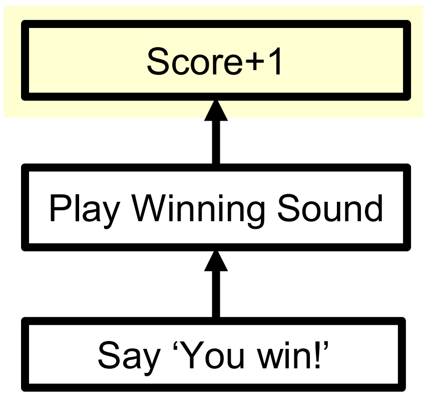
Testing and Debugging

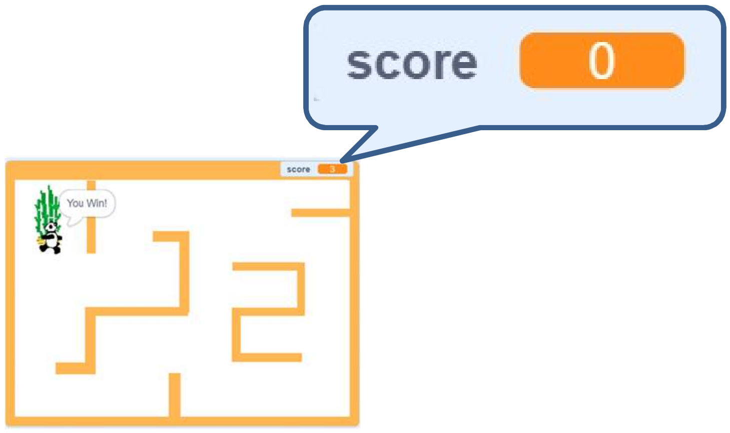
Try you best to move the Panda to the bamboo.
Don't forget to check the conditions constantly.



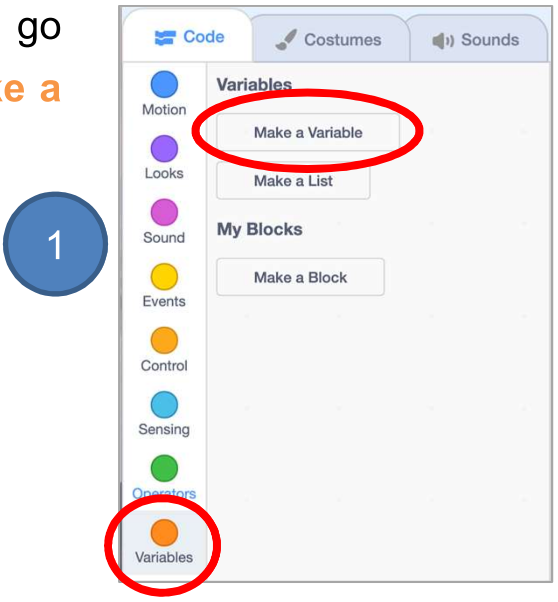
To Think and To Code: Add Score

Do you still remember what we get if the panda finds the bamboo? Yes! Now it's time to add score when you win.

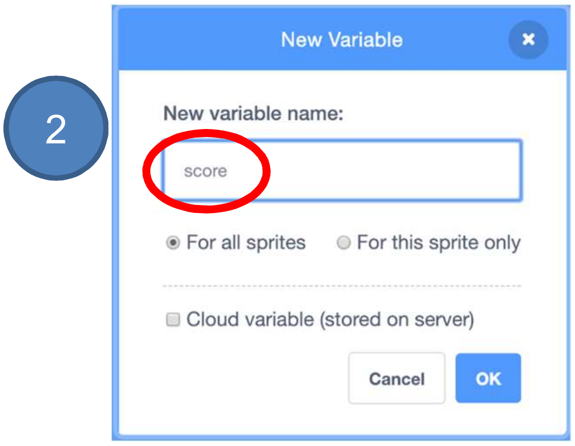




1. To add a variable for recording the score, go to the **"Variables"** drawer and click **"Make a Variable"**.



2. Name the new variable as **"score".**


New Variable

New variable name:

score

(● For all sprites) ○ For this sprite only

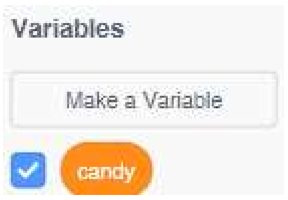
☐ Cloud variable (stored on server)

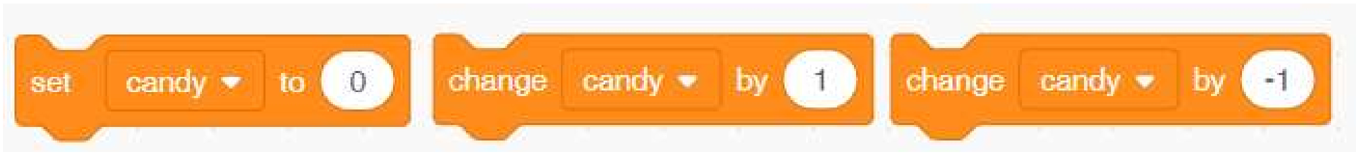
Cancel OK

Creating a Maze Game

Unplugged Activity: Variables

Teacher should prepare a box and some candies to run this game.

	Teacher takes out a box and writes "candy" on the box

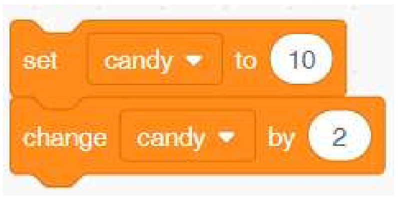
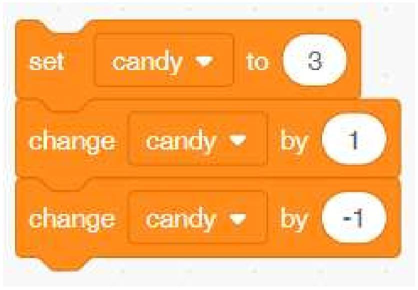
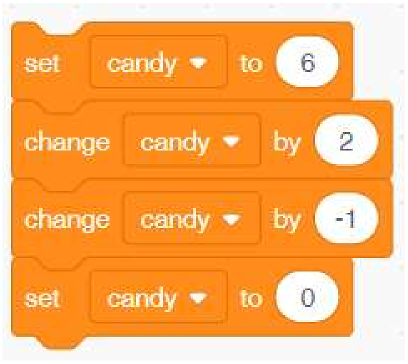


Game 1: You can choose one of them, then teacher will do the action.

1. Set candy to " 5"
2. Change candy by "-2"
3. Change candy by "2"

Students can choose one of them, then teacher will do the action.
Students: Set candy to " 5", teacher takes out all candies from the box and then put 5 candies in the box
Students: Change candy by "-2", teacher takes out 2 candies from the box
Students: Change candy by "2", teacher puts 2 candies in the box

Game 2: Guess how many candies in the box:

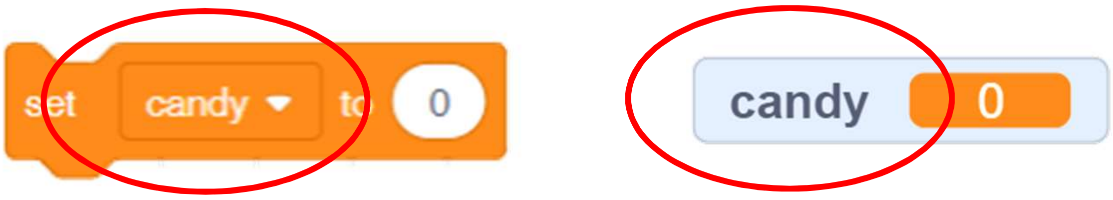
12	3	0

Creating a Maze Game

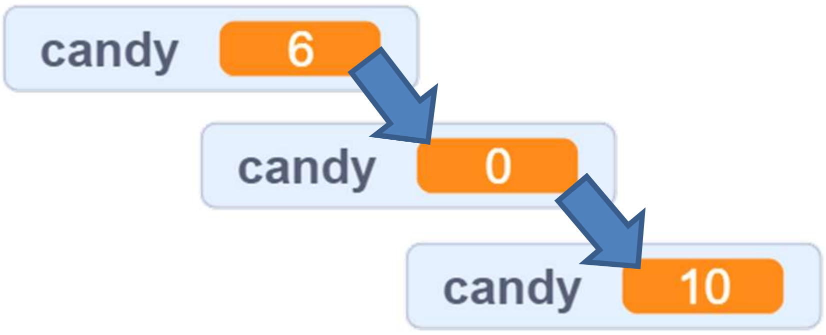
Knowledge Builds up: Variables

Variables are used to store values. Variables have the following properties.
Fill in the blank:

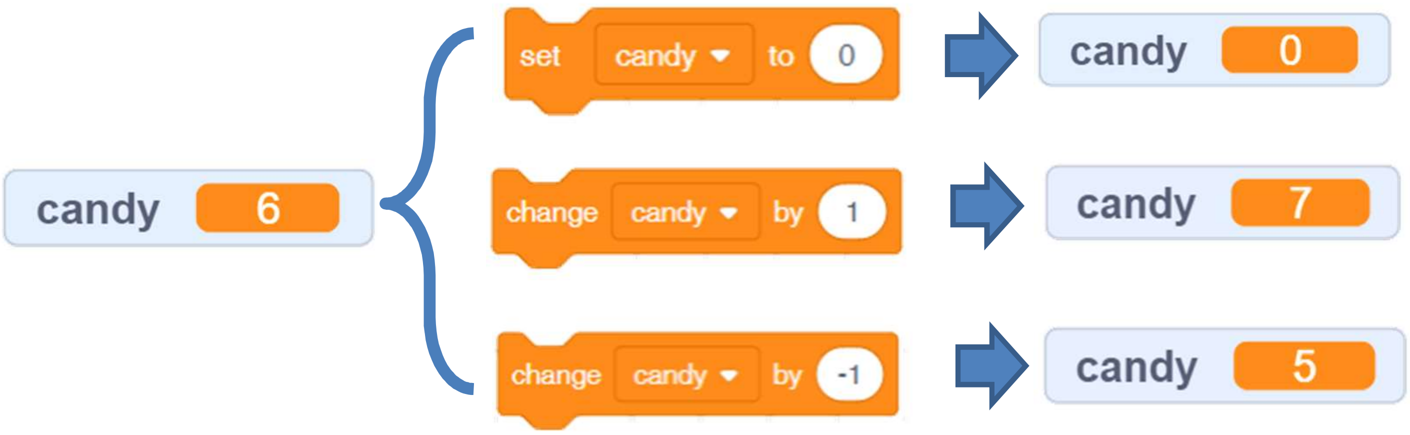
1. Variables have names.



2. A variable can only store one value at a time.



3. The value of the variable can be updated.

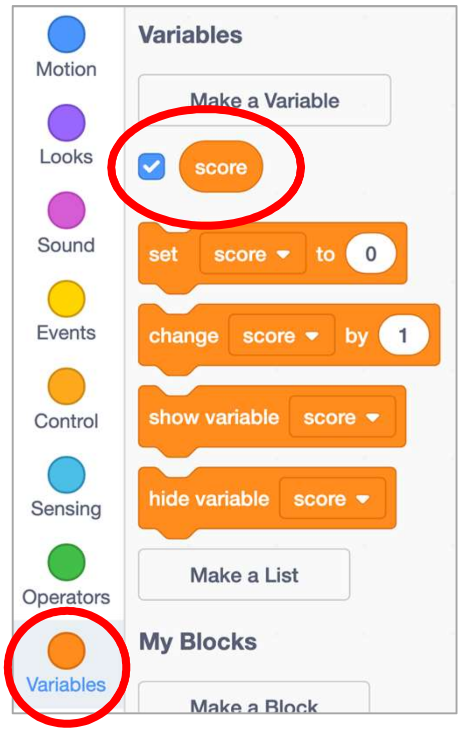


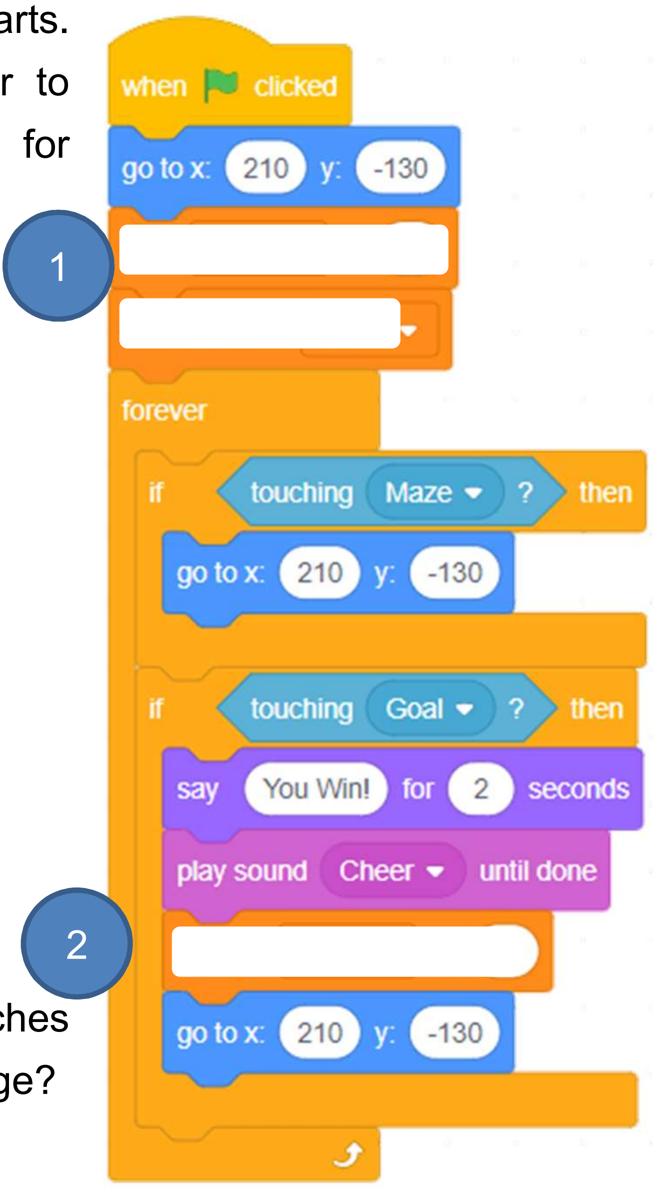
Creating a Maze Game

To Code: Add Score

See Appendix P.38

1. First of all, we need to show variable score (tick the score to show) and reset the score to 0 when the game starts. Hint: Check out the "Variables" Drawer to see which block should be added for resetting the score.

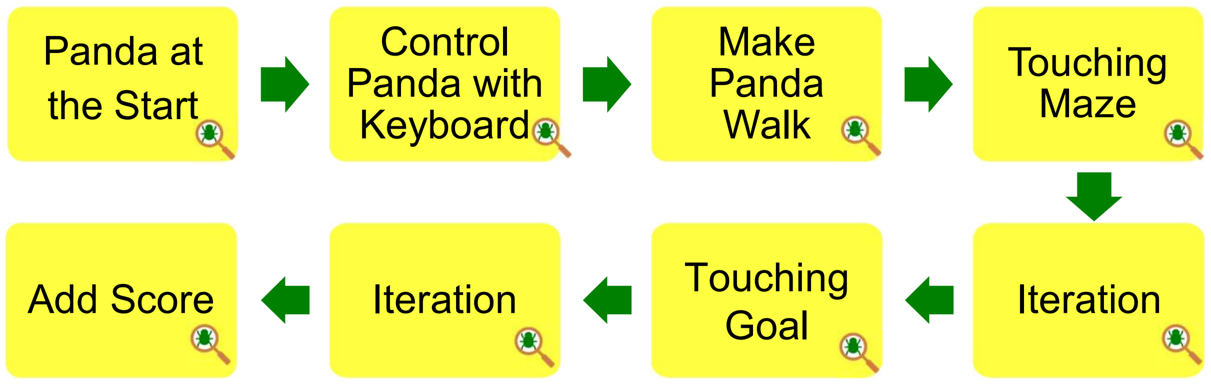




2. When the Panda sprite reaches the bamboo, what should the score change?

Testing and Debugging

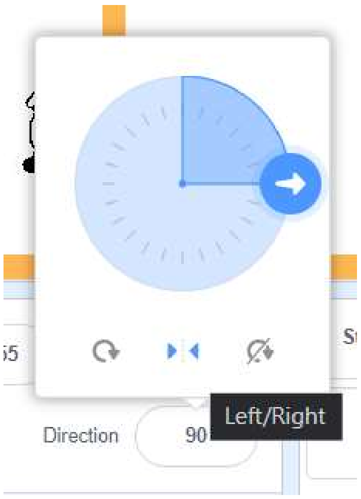
You did it! Play your maze game again to see everything works fine.

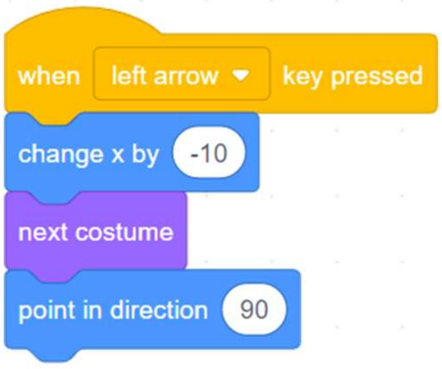


Program codes for reference:

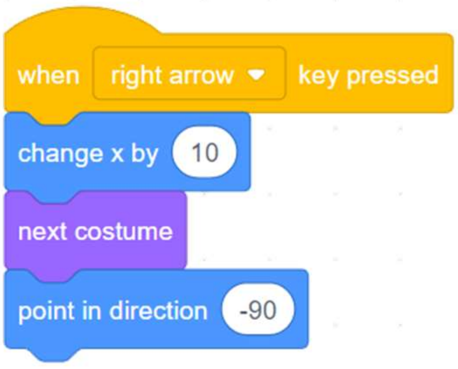
After completing the task, students can be lead to reflect on how to further enhance the program / maze game.
Some examples can be provided as an extended activities for more able students.
E.g. changing the direction of the panda's face when walking backwards, adding (bouncing obstacles), etc.

Changing the direction of the panda's face when walking backwards:



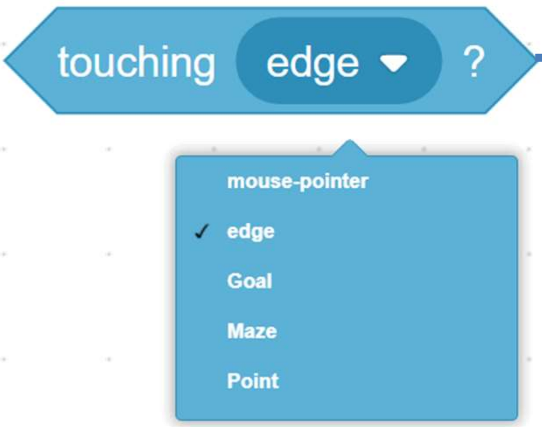


```
when [left arrow ▾] key pressed
change x by (-10)
next costume
point in direction (90)
```

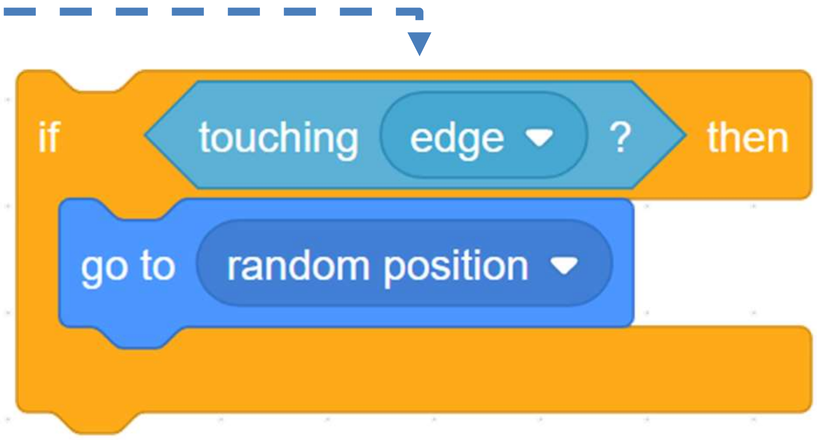


```
when [right arrow ▾] key pressed
change x by (10)
next costume
point in direction (-90)
```

Adding edge / bouncing obstacles:



```
<touching [edge ▾] ?>
```

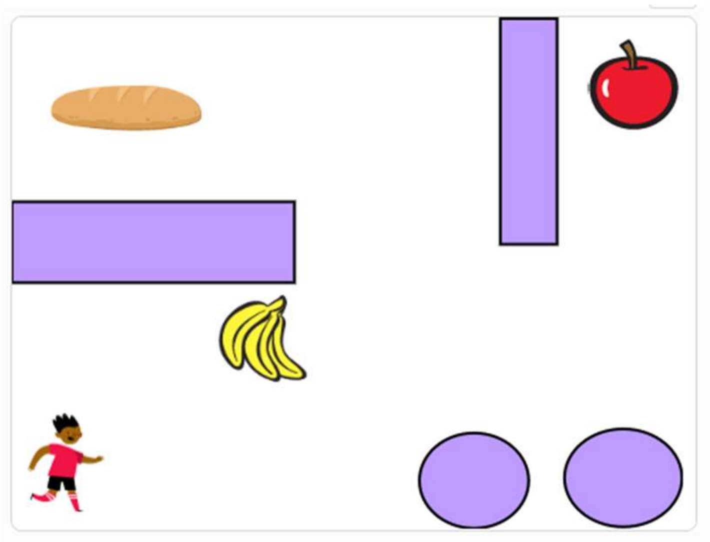


```
if <touching [edge ▾] ?> then
    go to [random position ▾]
```

Creating a Maze Game

To Create: Supermarket Maze

1. With your creativity, you are free to create another maze game, or you may take a look at the example project, Supermarket Maze.
2. There is another project similar to the maze game we just learnt: Control the boy sprite to find the bread in the supermarket maze. if the boy is “touching” the bread, then he says, “I find it!”
   Example: Supermarket Maze https://scratch.mit.edu/projects/731275755/



Write down your idea:

(e.g.) If the boy is touching banana then go back to the starting point.

1. If the boy is touching Apple, then go back to the starting point.

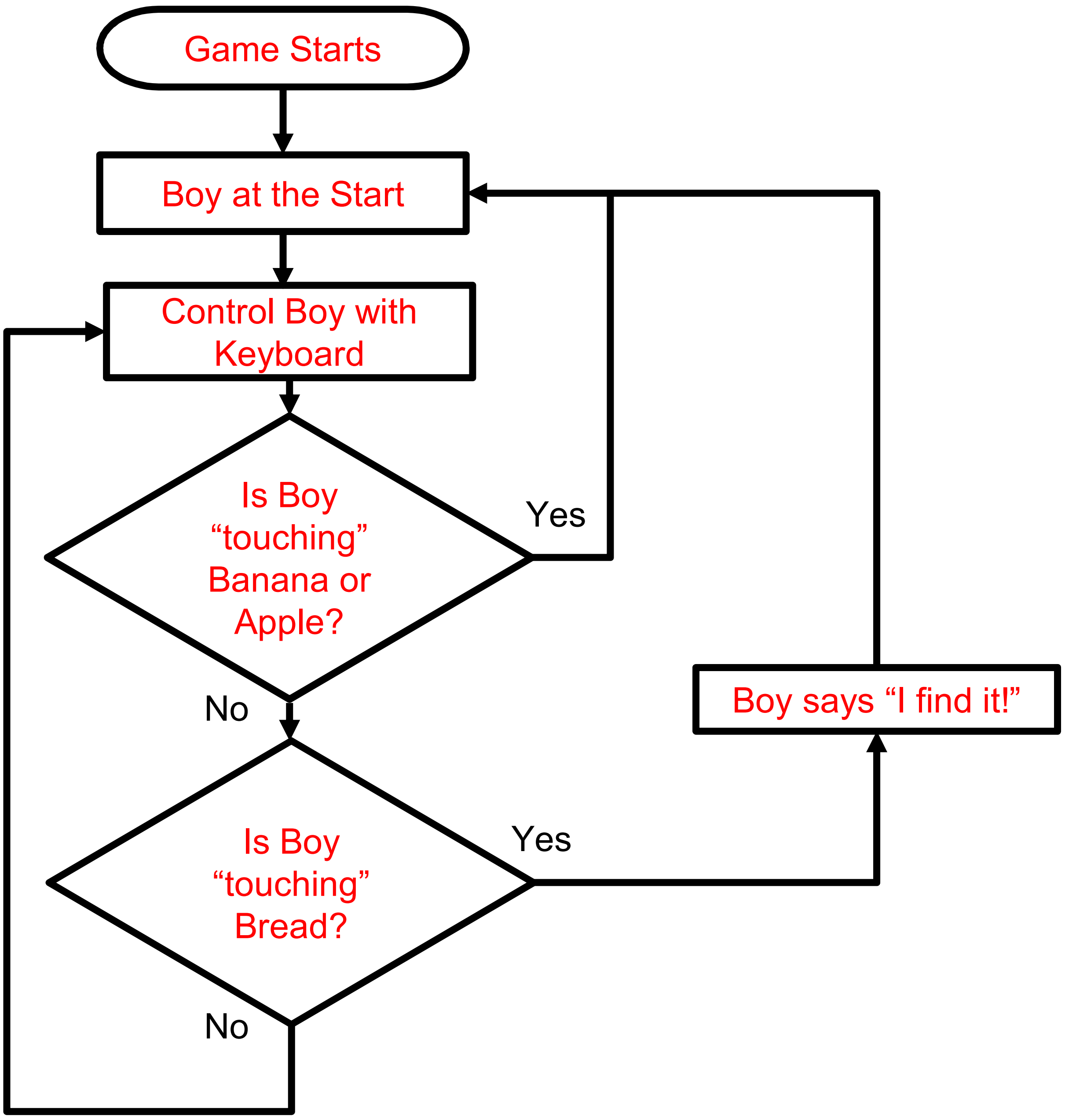
2. If the boy is touching Bread, then say I find it!

In addition to recreating a new game, students can be encouraged to add more traps, change the structure of the maze, or change the sprites in the panda game etc.

Creating a Maze Game

To Think and To Code

Based on what you have learnt in this unit, think about the sequence and algorithm design of your own Maze Game. Fill in the blank to complete the flow diagram .



With the flow diagram you design, program it to achieve your idea!

Creating a Maze Game

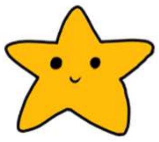
To Reflect: Two Stars and a Wish Worksheet

Name of Project: ___________ Name of Creator: ___________

Please write down two things that you like about this project.

1

"Two Stars and a Wish" is a reflection strategy designed for student feedback as peer- and self-assessment.
Teachers can guide students to give constructive feedback to their peers regarding their Scratch project - two positive (stars) and one hopeful (wish) reflection. Comments can be made on Scratch project's idea, features and aesthetic aspects etc.

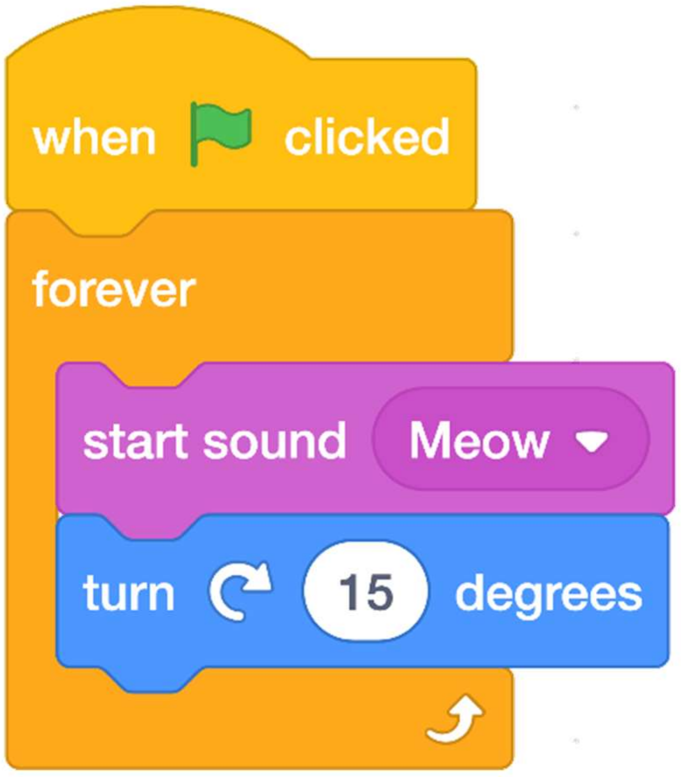


2

What is one thing you would like to add or change to make this project better?

Review Questions

1. What happens to the cat when you click the green flag?



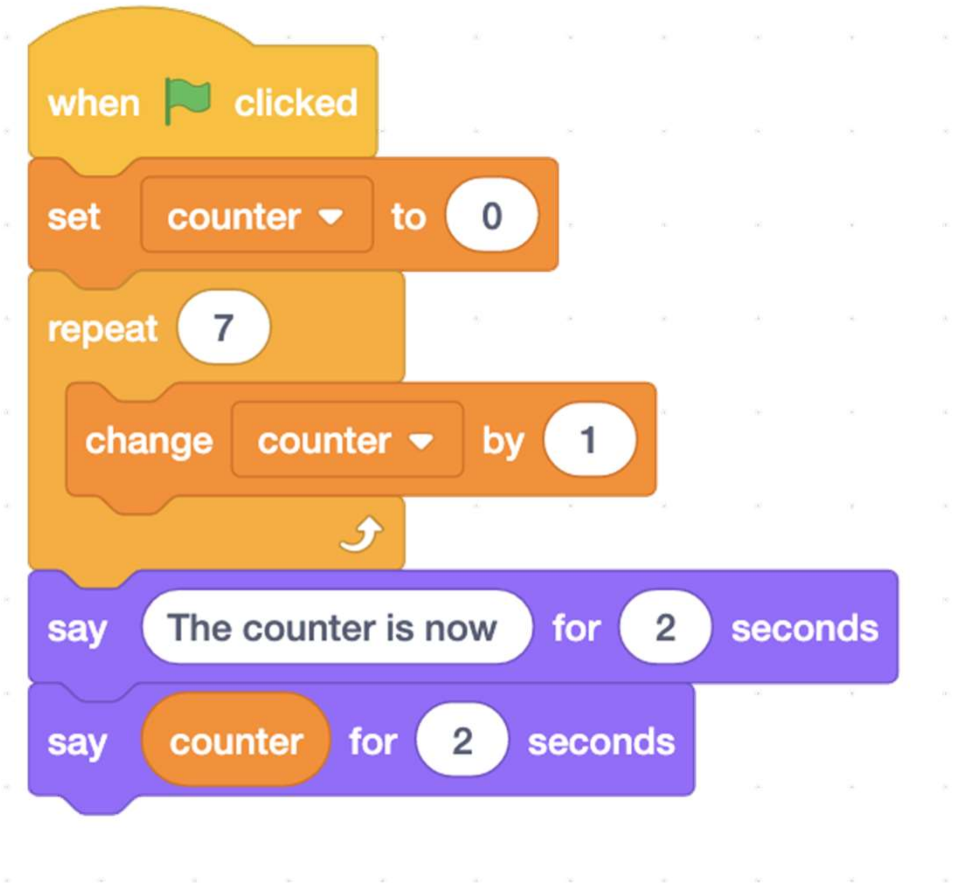
```
when ⚑ clicked
forever
    start sound Meow ▾
    turn ↻ 15 degrees
```

A. The cat meows once and turns 15 degrees once.
B. The cat meows repeatedly forever but never turns.
C. The cat meows once but then turns 15 degrees repeatedly (spins clockwise) without meowing.
D. The cat meows repeatedly while turning 15 degrees (spins clockwise) repeatedly.

(Answer: D)

Review Questions

2. What happens when the code blocks below run?



```
when 🏁 clicked
set counter ▾ to 0
repeat 7
    change counter ▾ by 1
↻
say The counter is now for 2 seconds
say counter for 2 seconds
```

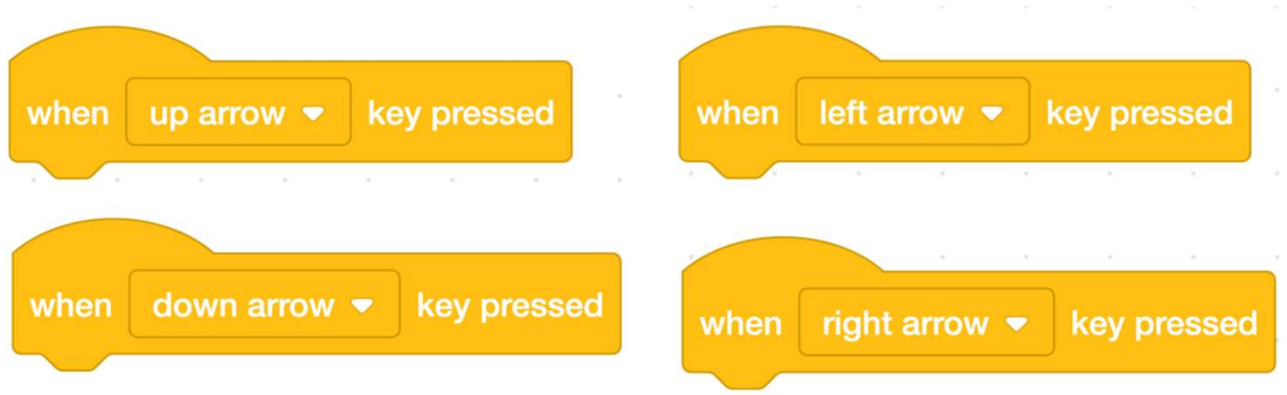
A. The Sprite says “The counter is now” and then says“1”, “2”, “3”, “4”, “5”, “6”, and “7” for 2 seconds.
B. The Sprite says “The counter is now” and then says “7” for 2 seconds.
C. The Sprite says “The counter is now” and then says “counter”.
D. The Sprite says “The counter is now” and then says “7” seven times for 2 seconds.

(Answer: B)

Creating a Maze Game

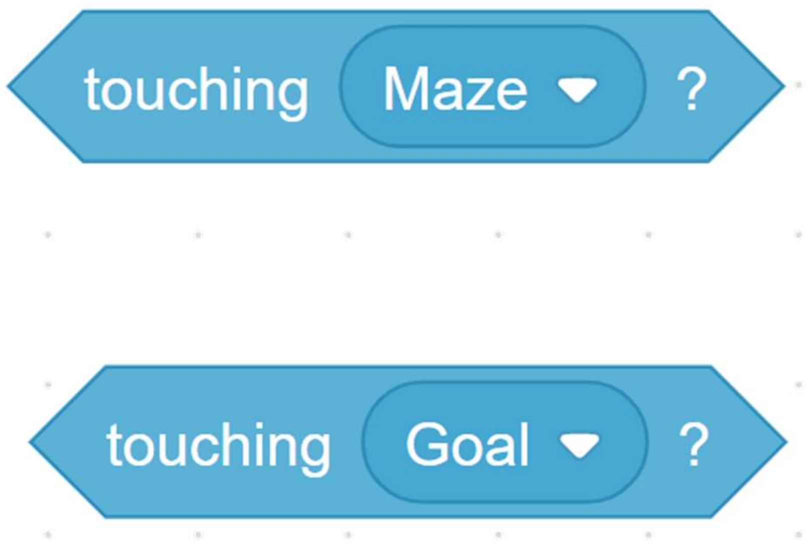
Revision on Key Features

Key pressed events:



```
when [up arrow ▾] key pressed
when [left arrow ▾] key pressed
when [down arrow ▾] key pressed
when [right arrow ▾] key pressed
```

“Touching” blocks:

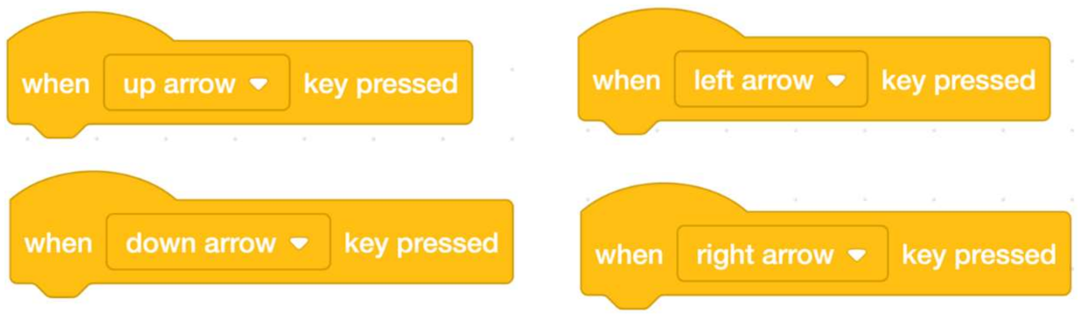


```
<touching (Maze ▾) ?>
<touching (Goal ▾) ?>
```

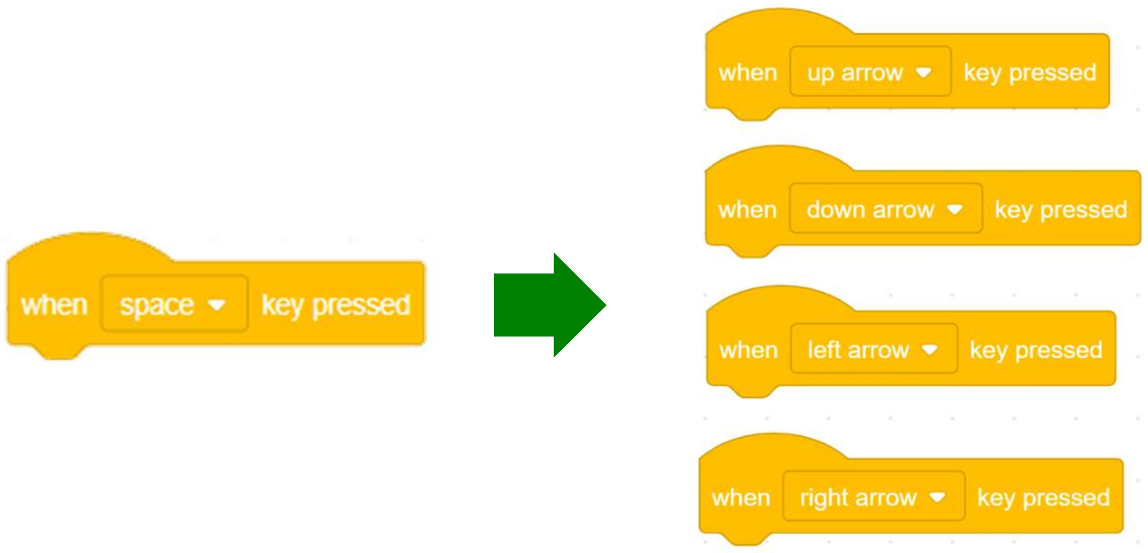
Creating a Maze Game

Revision on Key Concepts & Practices

Events: We use event blocks (keyboard events) to trigger Scratch to take actions (sprite movement).

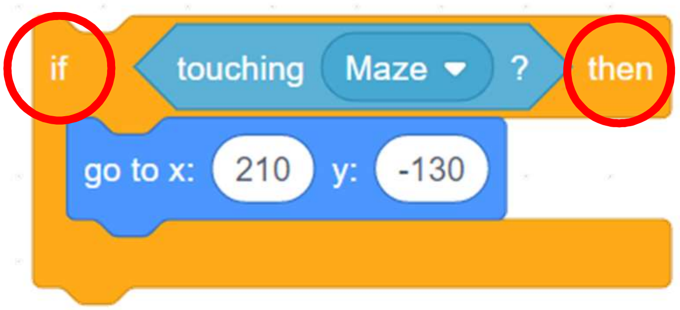

when up arrow ▾ key pressed
when left arrow ▾ key pressed
when down arrow ▾ key pressed
when right arrow ▾ key pressed

Reuse and Remix programs/codes: The reuse and remix of the works of other programmers are crucial in the online communities of Scratch. We can reuse and remix the codes of one keyboard event and use them for the rest of arrow key pressed events.


when space ▾ key pressed
when up arrow ▾ key pressed
when down arrow ▾ key pressed
when left arrow ▾ key pressed
when right arrow ▾ key pressed

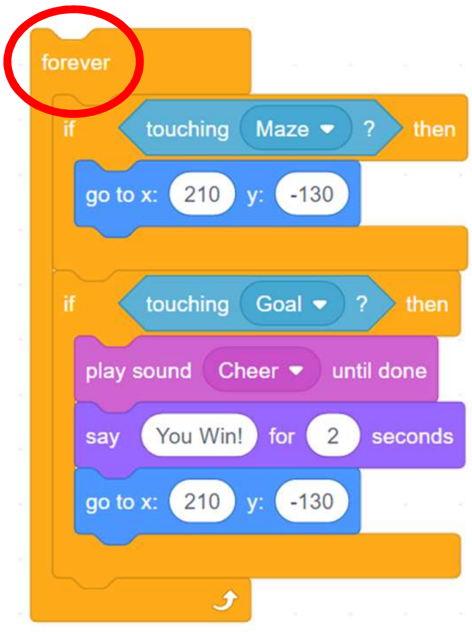
Revision on Key Concepts & Practices

Branching/Selection: We use conditional statements in programming to enable computers to make decisions. Conditionals always have an "If" part, which tells the program in the "Then" part what to do when the condition is true.



```
if <touching (Maze ▾) ?> then
    go to x: (210) y: (-130)
```

Iteration - Forever: Iteration is repeating a process in order to produce a sequence of outcomes. Forever blocks can trigger iteration in Scratch.

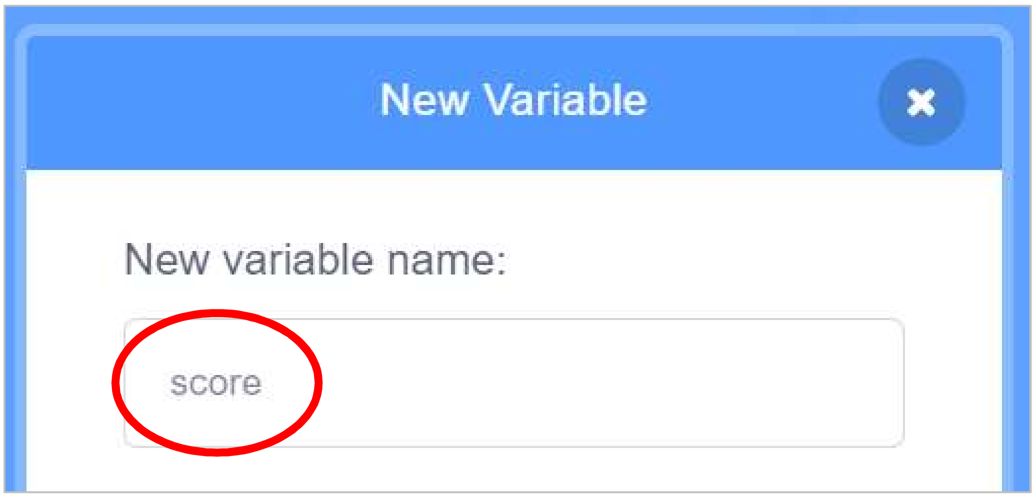


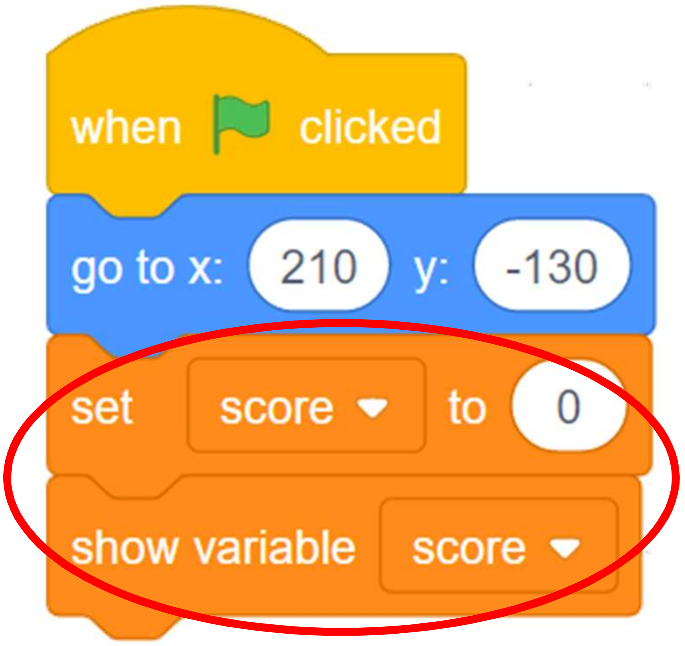
```
forever
    if <touching (Maze ▾) ?> then
        go to x: (210) y: (-130)
    if <touching (Goal ▾) ?> then
        play sound (Cheer ▾) until done
        say (You Win!) for (2) seconds
        go to x: (210) y: (-130)
```

Creating a Maze Game

Revision on Key Concepts & Practices

Variables: In programming, variables are used to store values. It has a name, can only store one value at a time and be updated. For example, We can create a variable called “score" to store the score of a game.

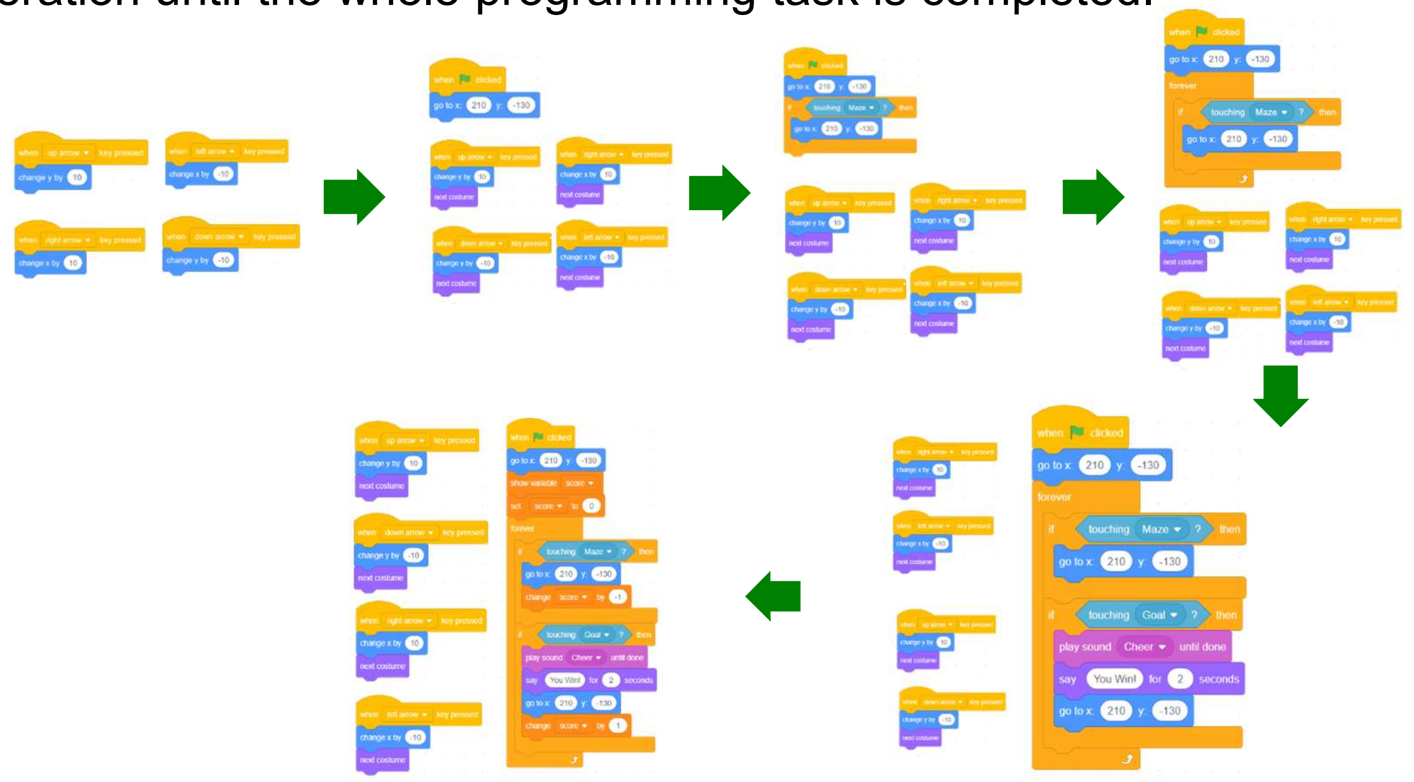




Creating a Maze Game

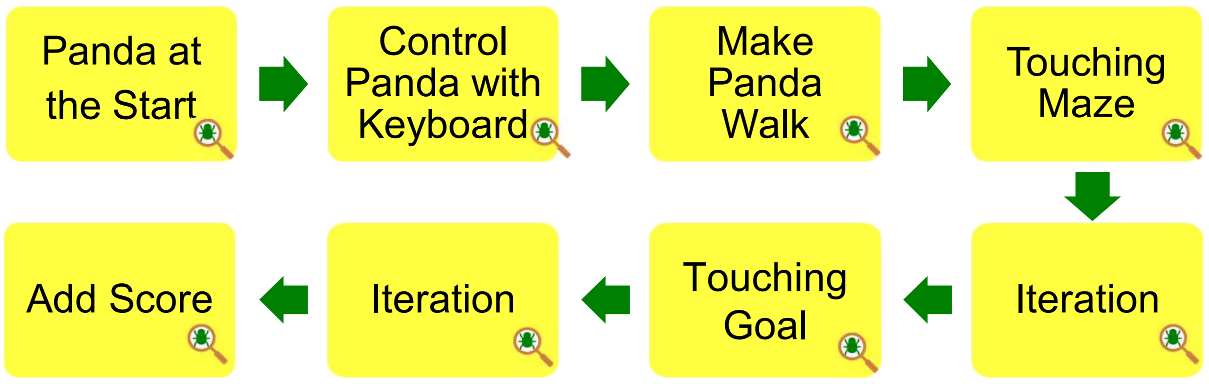
Revision on Key Concepts & Practices

Being incremental and iterative: to work out a sub-task as an iteration, try it out, then work out another sub-task in one more iteration until the whole programming task is completed.



```
when [up arrow] key pressed
change y by (10)
next costume

when [down arrow] key pressed
change y by (-10)
next costume

when [right arrow] key pressed
change x by (10)
next costume

when [left arrow] key pressed
change x by (-10)
next costume

when green flag clicked
go to x: (210) y: (-130)
show variable [score]
set [score] to (0)
forever
    if <touching [Maze] ?> then
        go to x: (210) y: (-130)
        change [score] by (-1)
    if <touching [Goal] ?> then
        play sound [Cheer] until done
        say (You Win!) for (2) seconds
        go to x: (210) y: (-130)
        change [score] by (1)
```

Testing and Debugging: Testing a computer program is the process of checking if it can produce results as designed. Debugging a computer program is the process of finding out ways to revise the program so that the bugs can be removed.



Appendix

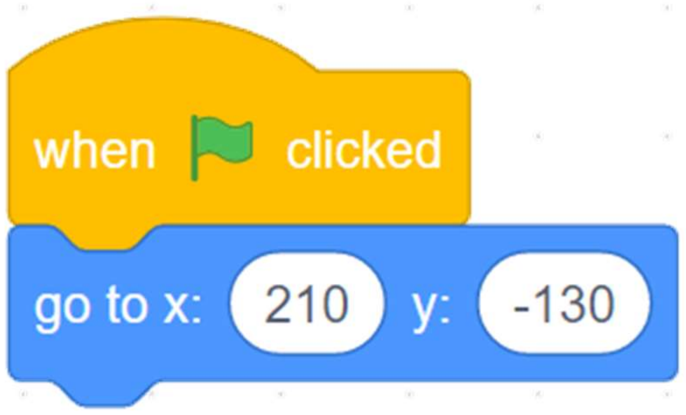
Operation Manual

Creating a Maze Game

To Code: Panda at the Start

See Teacher Guide P.5

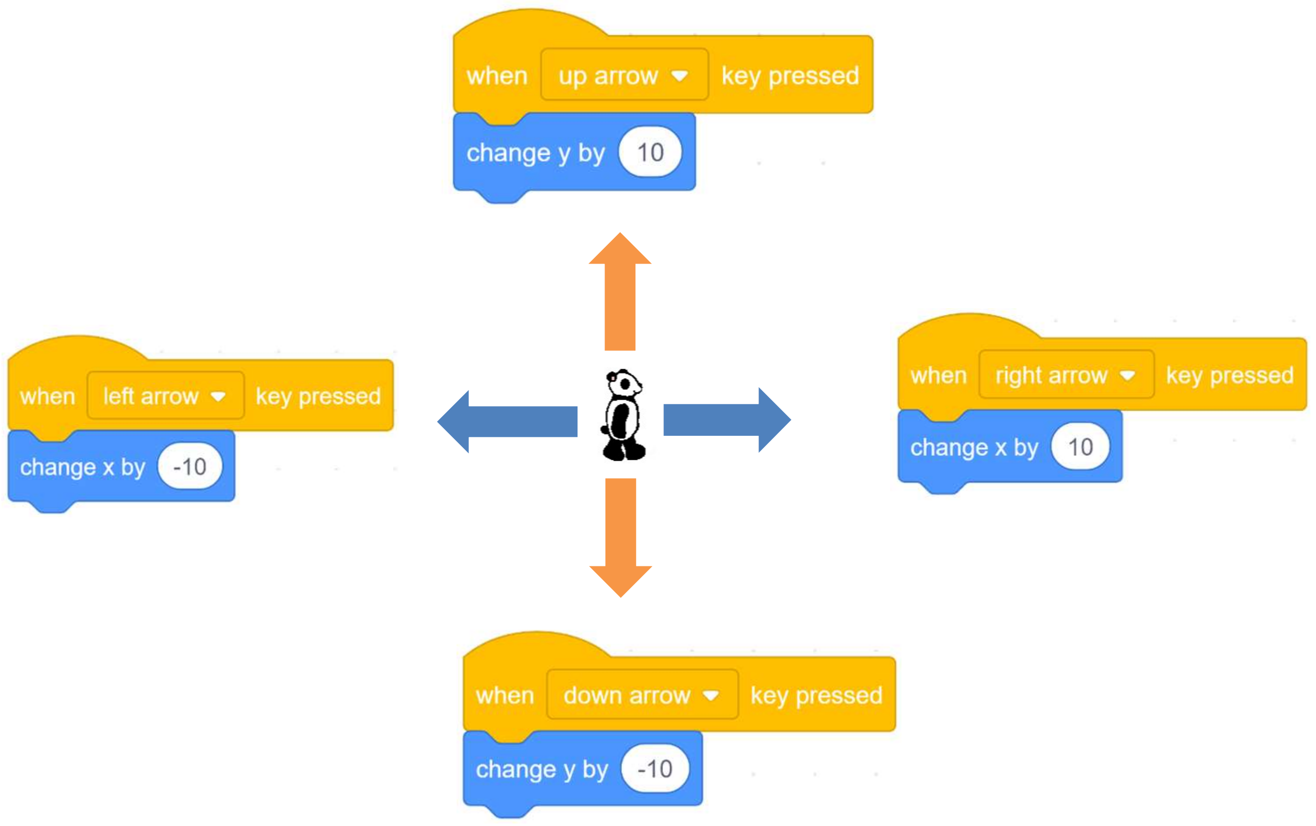
Make your sprite go to your desired starting coordinates when the green flag is clicked.



```
when 🏴 clicked
go to x: 210 y: -130
```

To Code: Control Panda with Keyboard

See Teacher Guide P.6



```
when up arrow key pressed
change y by 10
```

```
when left arrow key pressed
change x by -10
```

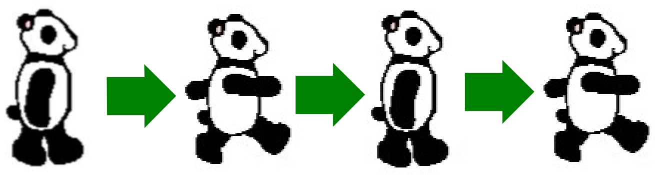
```
when right arrow key pressed
change x by 10
```

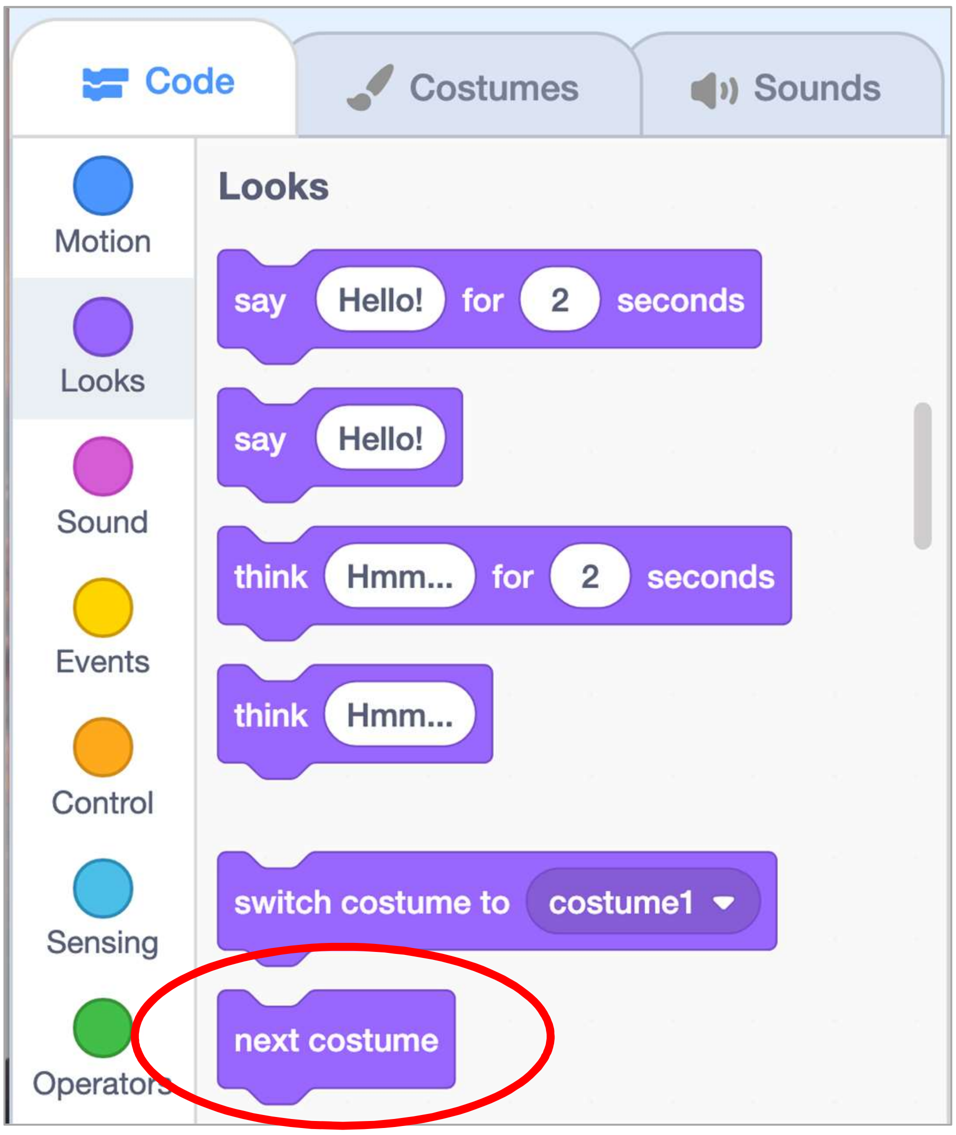
```
when down arrow key pressed
change y by -10
```

To Think and To Code: Make Panda Walk

Use the "**next costume**" block to make the panda look like it is walking.

See Teacher Guide P.7





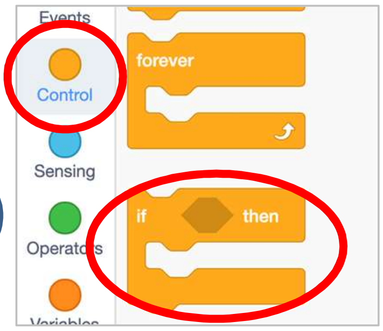
Creating a Maze Game

To Code: Touching Maze

See Teacher Guide P.12

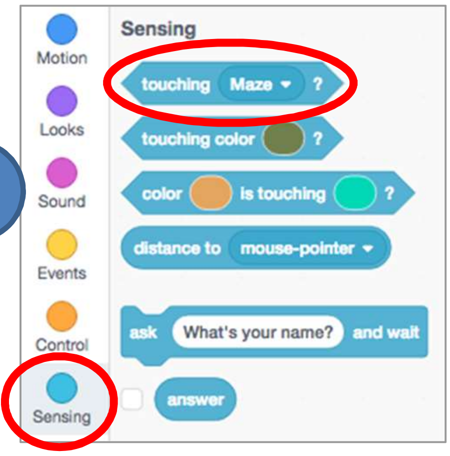
1. In this game, we want to send the Panda sprite to the starting point if it touches the walls of the maze. To do this, pull out an "**if-then**" block from the "**Control**" drawer to complete this task.

1



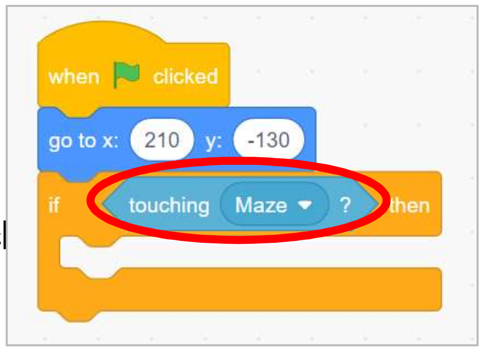
2. We want the Panda sprite to go to the start if it touches the Maze sprite, so also pull out a "**touching Maze**" block from the "**Sensing**" drawer.

2



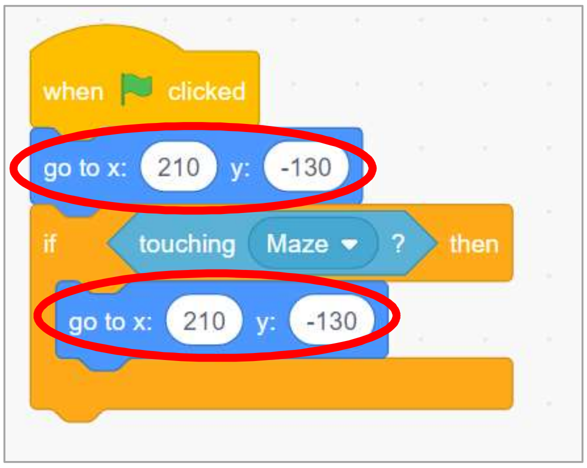
3. Put the "**touching Maze**" block in the empty slot of the " **if-then**" block and put that combined block under the "**when green flag clicked**" block.

3



4. Copy and paste the "**go to x:# y:#**" bloc
inside the "**if-then**" block.

4



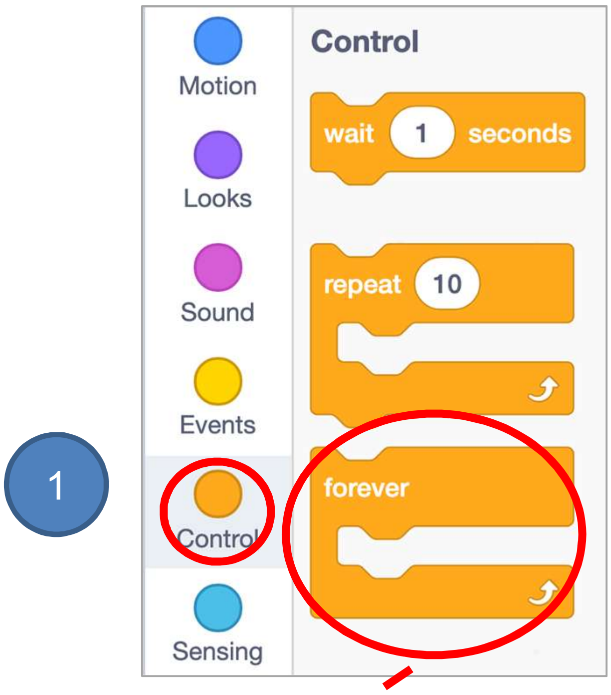
Creating a Maze Game

To Think and To Code: Iteration

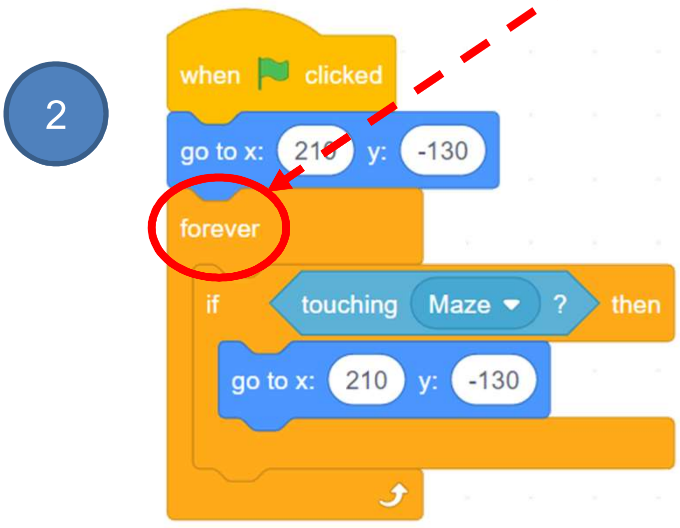
See Teacher Guide P.14

The “**forever**” block allows the program to constantly check for the conditions you are testing.

1. Add the “**forever**” block from the “**Control**” drawer.



2. Put the **if touching Maze** block inside the **forever** block.



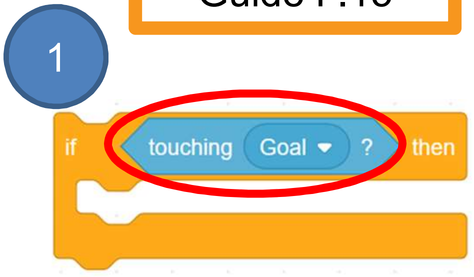
```
when 🏳 clicked
go to x: 210 y: -130
forever
    if touching Maze ? then
        go to x: 210 y: -130
```

Creating a Maze Game

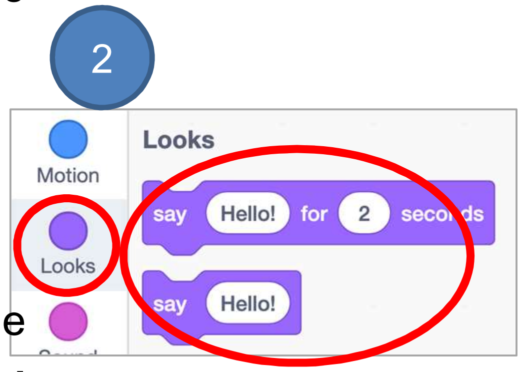
To Code: Touching Goal

See Teacher Guide P.15

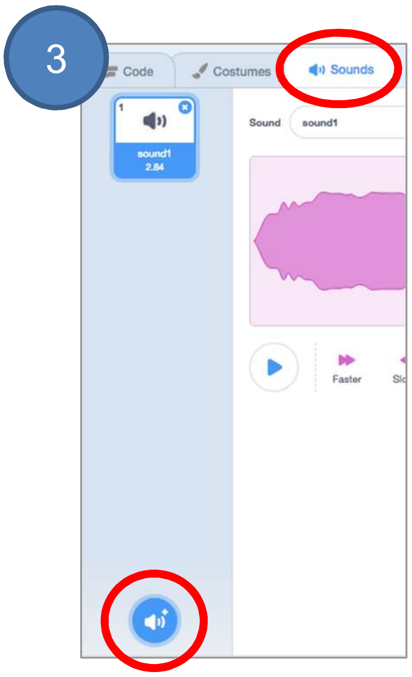
1. Add the appropriate additional "**if-then**" block if the Panda sprite touches the goal, it says "You win!" and plays a sound.



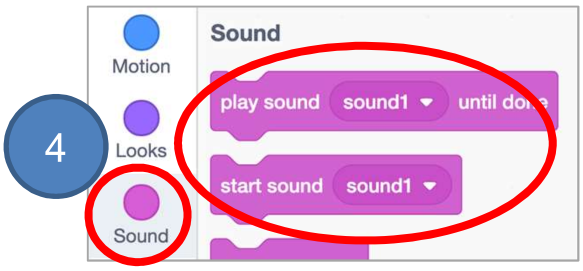
2. To make your sprite say something, go to the "**Looks**" drawer.



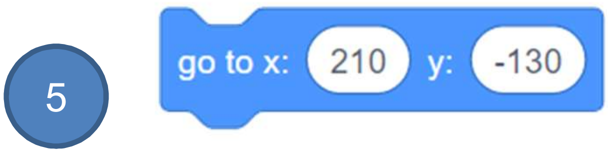
3. To add a sound to your project, go to the "**Sounds**" tab. Click on the **Choose a Sound** icon at the bottom left to select a sound.



4. To make your sprite play the sound you found, go to the "**Sound**" drawer of "**Code**" tab.



5. Finally send the Panda sprite back to its starting position.

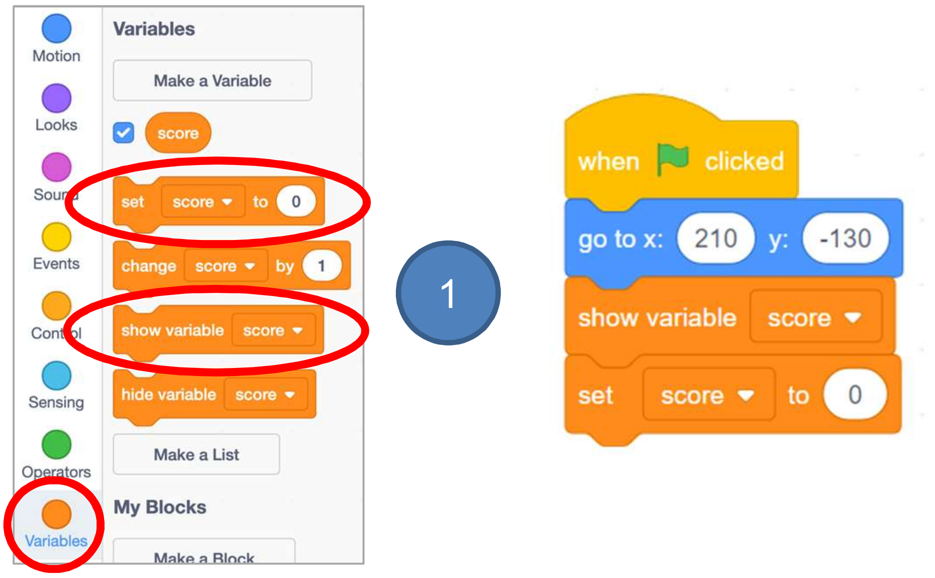


Creating a Maze Game

To Code: Add Score

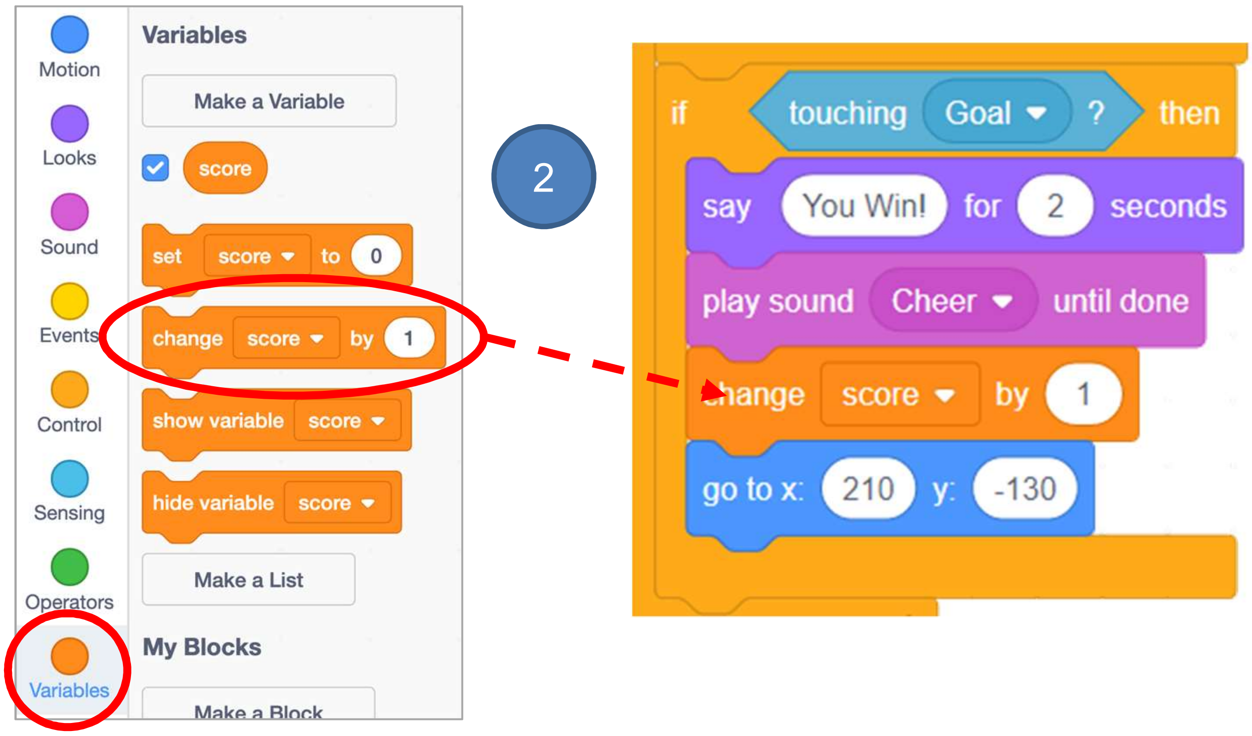
See Teacher Guide P.20

1. Drag “**set score to 0**” and “**show variable score**” blocks to “**when the green flag is clicked**” block.



```
when green flag clicked
go to x: 210 y: -130
show variable score
set score to 0
```

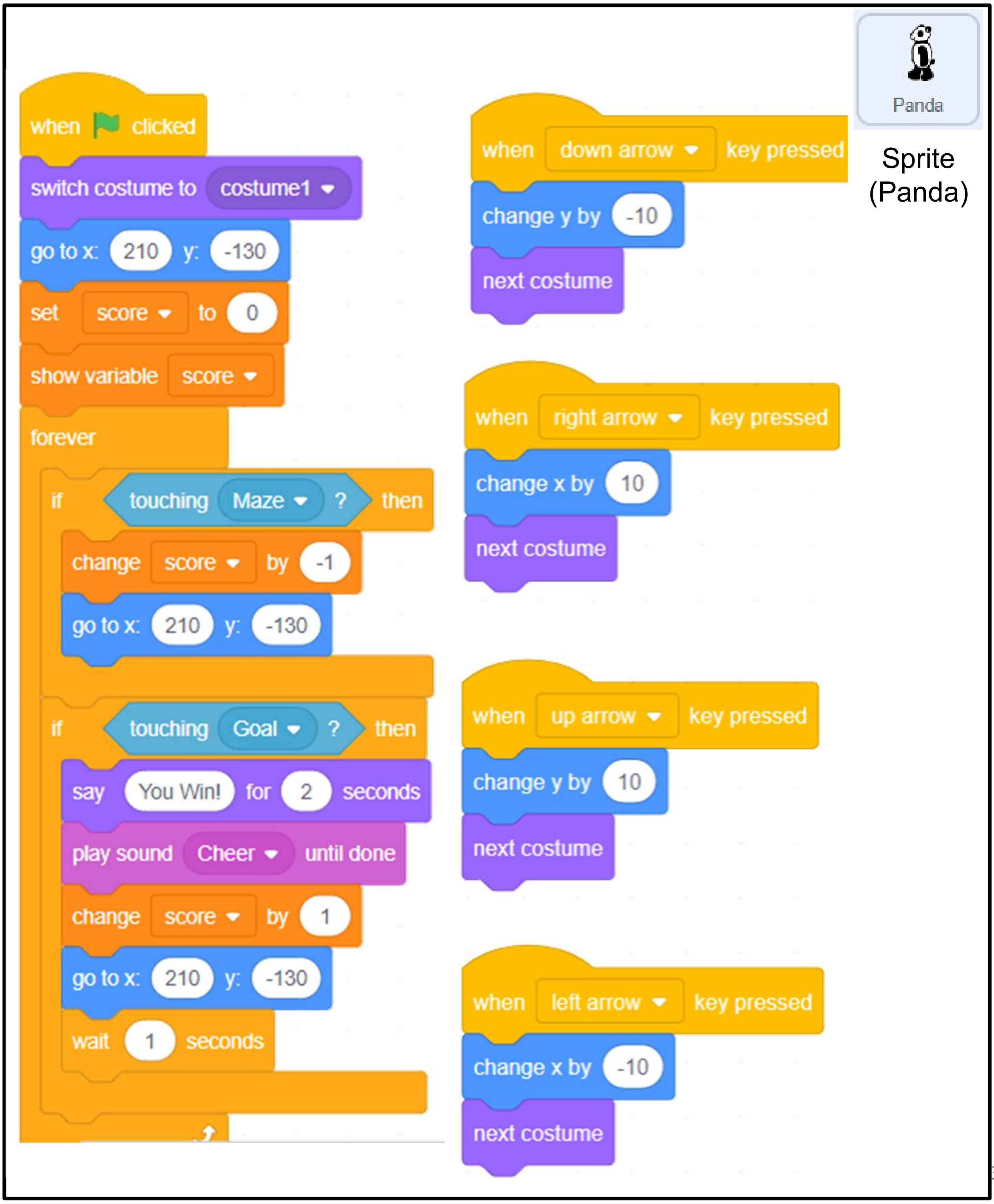
2. Change the score by 1 when the Panda sprite reaches the bamboo. Put “**change score by 1**” under “**if touching goal**” block.



```
if touching Goal ? then
    say You Win! for 2 seconds
    play sound Cheer until done
    change score by 1
    go to x: 210 y: -130
```

Creating a Maze Game

Program Codes: Maze Game with Panda



```
when 🏁 clicked
switch costume to costume1
go to x: 210 y: -130
set score to 0
show variable score
forever
    if touching Maze ? then
        change score by -1
        go to x: 210 y: -130
    if touching Goal ? then
        say You Win! for 2 seconds
        play sound Cheer until done
        change score by 1
        go to x: 210 y: -130
        wait 1 seconds
```

```
when down arrow key pressed
change y by -10
next costume
```

```
when right arrow key pressed
change x by 10
next costume
```

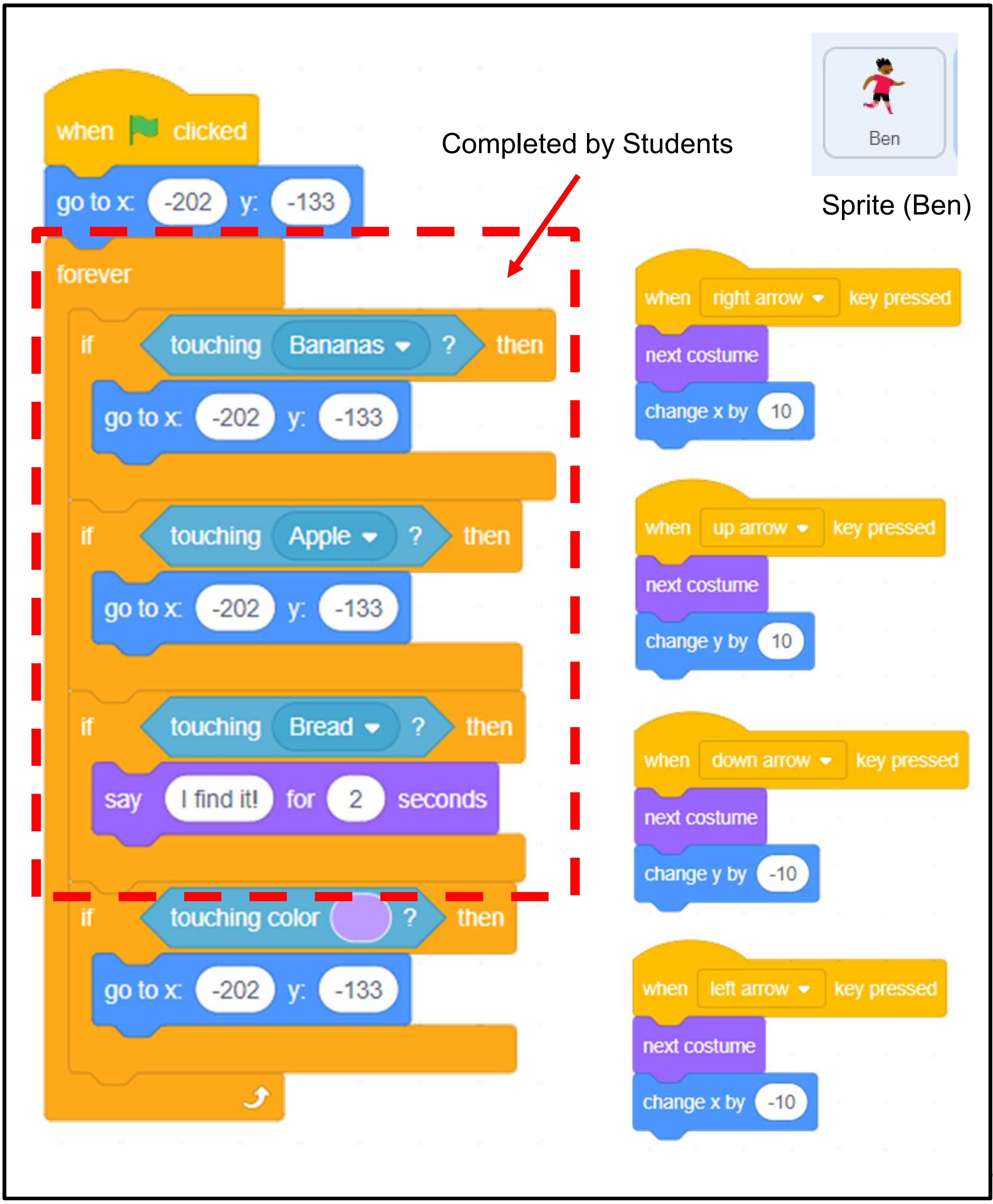
```
when up arrow key pressed
change y by 10
next costume
```

```
when left arrow key pressed
change x by -10
next costume
```

Sprite (Panda)

Creating a Maze Game

Program Codes: Supermarket Maze



```
when green flag clicked
go to x: -202 y: -133
forever
    if touching Bananas ? then
        go to x: -202 y: -133
    if touching Apple ? then
        go to x: -202 y: -133
    if touching Bread ? then
        say I find it! for 2 seconds
    if touching color ? then
        go to x: -202 y: -133

when right arrow key pressed
next costume
change x by 10

when up arrow key pressed
next costume
change y by 10

when down arrow key pressed
next costume
change y by -10

when left arrow key pressed
next costume
change x by -10
```

Unit 6: Creating a Maze Game with micro:bit（Extension Unit）Teacher Guide

Content

Unit 6: Creating a Maze Game with micro:bit (Extension Unit) Teaching Plan

Prior Knowledge

Students should demonstrate a basic understanding of Scratch programming. They should know the skills of keyboard pressed events and how to use variable for scoring.

Learning Objectives

1. Demonstrate an understanding of basic functions and features of a micro:bit;
2. Understand the concept of a system and automation through interacting with physical objects;
3. Develop an initial understanding of Internet-of-Things (IoT) (Sensing-Reasoning-Reacting);
4. Enable student to explore how they could use micro:bit.

Learning Elements

Computational Thinking Concepts and Practices:

Key Learning Elements	Items
Abstraction	Express the algorithm, Modularization
Algorithm	Problem Solving Procedures: Problem identification, Problem analysis, Algorithm design, Programming Basic Programming Constructs: Sequence, Branching / Selection, Iteration Data Processing: Variable Coding Concept and Practices: Design, Reuse and Remix programs / codes, Testing and Debugging
Interacting with Physical Objects	Use of micro:bit Concept of a system and automation Extended learning: Internet-of-Things (IoT) (Sensing-Reasoning-Reacting)

Coding Skills:

1. Add and use micro:bit blocks;
2. Use iteration - forever blocks;
3. Reuse and remix programs / codes;
4. Apply Testing and Debugging in completing each task of this unit.

Others (including Attitude):

1. Develop interest in programming and interacting with physical object;
2. Show perseverance and positivity in hands-on activity, testing and debugging;
3. Inspire students to be creative and innovative to enhance their maze game projects with micro:bit.

Lesson Plan: This unit consists of 3 lessons of 35 minutes.

Teacher Preparation for the Lesson:
1. Teachers may prepare few completed paper models with micro:bit installed and some computers with Scratch Link ready, so that students can play at the beginning of the lesson.
2. Prepare one micro:bit for each student and label the name of each micro:bit for easy reference.
3. Reference: *How to find the name of your micro:bit:* https://support.microbit.org/support/solutions/articles/19000067679-how-to-find-the-name-of-your-micro-bit

Pre-task for Students: Make the Joystick at home. Ask students to follow the student guide, with handicraft materials and domestic reusables (e.g. paper roll) to make the joystick at home, and bring it back to school for the lesson.

Lesson 1

Time	Activity
5 mins	Introduce the unit focus which is about an initial understanding on Internet of Things (IoT) ("Sensing-Reasoning-Reacting").
15 mins	**To Play** 1. Give each student (or in group) a micro:bit. 2. Guide students to connect their joystick with micro:bit to play the game. 3. Play the Scratch project Maze Game with micro:bit https://scratch.mit.edu/projects/734787236/ 4. Ask them to control the sprite: Not only use keyboard (arrow keys) but also use micro:bit to do so. 5. Ask them which control method, use keyboard or joystick, they prefer and why.
10 mins	**To Think** 1. Ask students to complete the exercise to understand the differences between the two maze game projects that with or without micro:bit. 2. Match the joystick direction and the sprite reaction. Ask students why they can use the joystick to control the sprite. 3. Explain that micro:bit plays a big role in this unit as it has an accelerometer.
5 mins	**To Learn** 1. Introduce the features of microbit. 2. Ask students how micro:bit interacts with Scratch so you can make this game work.

Lesson 2 and 3

Time	Activity
15 mins	**To Code** 1. Remix the completed maze game in Unit 5 as the template for this unit. https://scratch.mit.edu/projects/722154863 2. Prepare the environment, such as bluetooth and Scratch link. 3. Add micro:bit extension, connect it to Scratch and complete the coding with the help of student guide. 4. Causal reasoning: Students should prepare the environment well and make everything connected or they will fail to play the game/system.
5 mins	**To Play (Maze Game with Data Analysis)** 1. Maze Game with Data Analysis https://scratch.mit.edu/projects/734745981/ 2. Tell them that this advanced maze game will count your steps and generate a bar graph at the end.
15 mins	**To Code** 1. Read the coding of this advanced maze game. 2. Variables: Counting front/back/left/back steps. 3. Pen: Draw the bar graph with those counting steps.
15 mins	Before “To Create”, conclude the task, give explanation of Internet of Things (IOT), give simple daily life examples to enhance students’ awareness on the use of the technology.
10 mins	**To Create** 1. Ask students to make use of what they learnt in this unit and draw something new on the student guide. 2. Ask questions like: “How will you make use of the accelerometer on micro:bit to create something cool and interact with physical objects?”
10 mins	**Review of Student Learning** Revision on the key features of micro:bit, key concepts & practices learnt

Creating a Maze Game with micro:bit

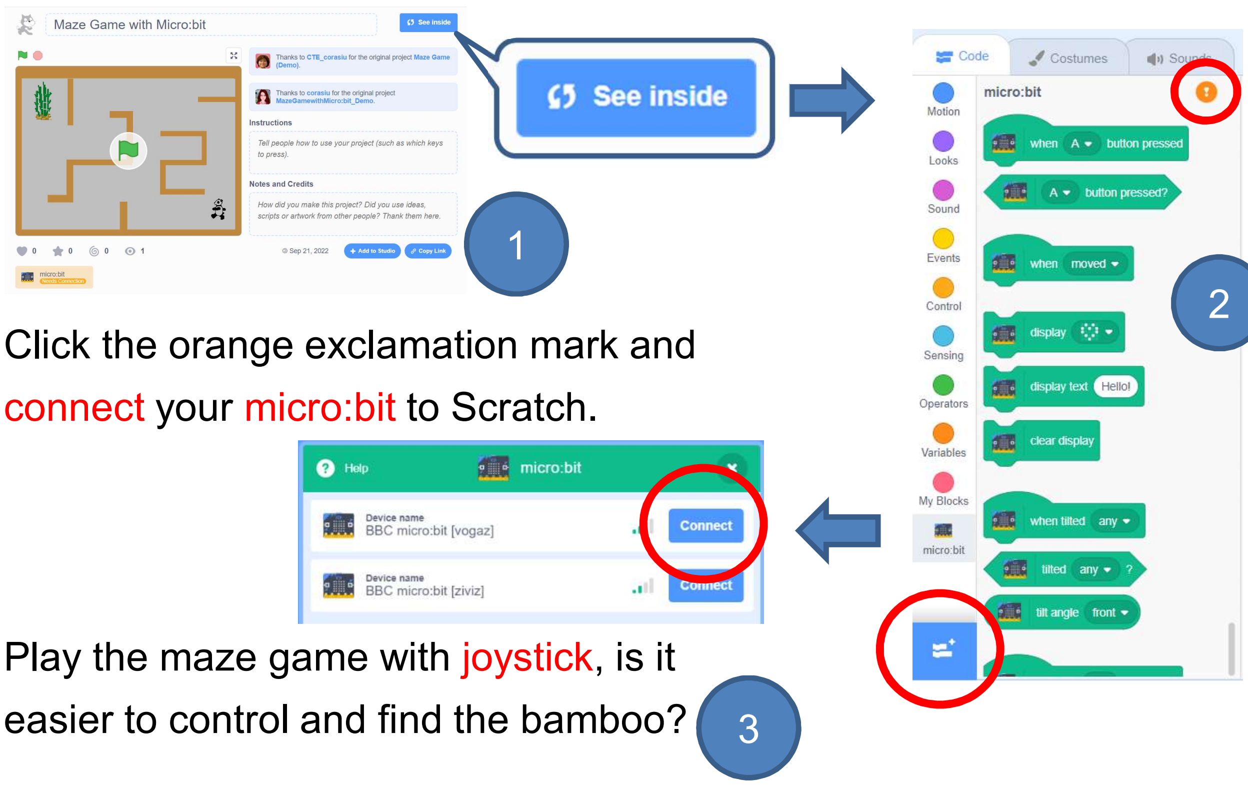
Let's create a maze game with micro:bit to control the sprite.

This game is similar to Unit 5. However, besides keyboard, you will control the panda sprite with a joystick (with micro:bit inside) this time.

To Play

Prepare your computer, make sure you have Scratch Link installed and running, and bluetooth is on. (Teacher may also provide you a joystick with micro:bit inside to play with).

1. Open https://scratch.mit.edu/projects/734787236/ and click **"See inside".**



2. Click the orange exclamation mark and connect your micro:bit to Scratch.

3. Play the maze game with joystick, is it easier to control and find the bamboo?

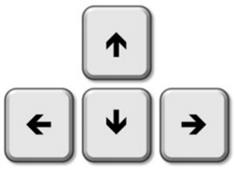
Creating a Maze Game with micro:bit

To Think

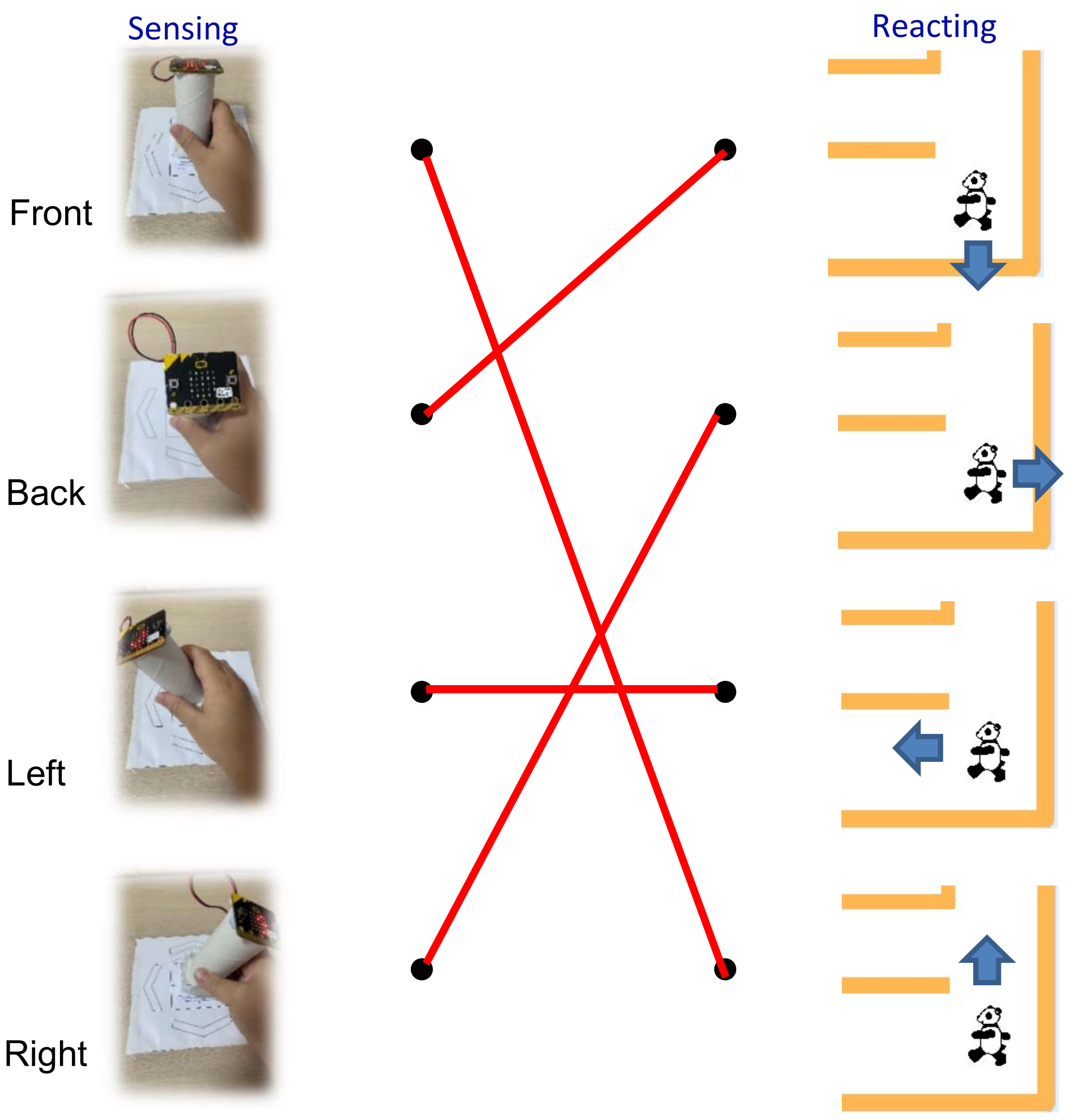
In this unit, the Maze Game looks a bit different. Mark the changes as follows:

1. How can you control the panda sprite this time? (You may tick ☑ more than one box.)

✓ Keyboard　　✓ Micro:bit　　❑ Mouse



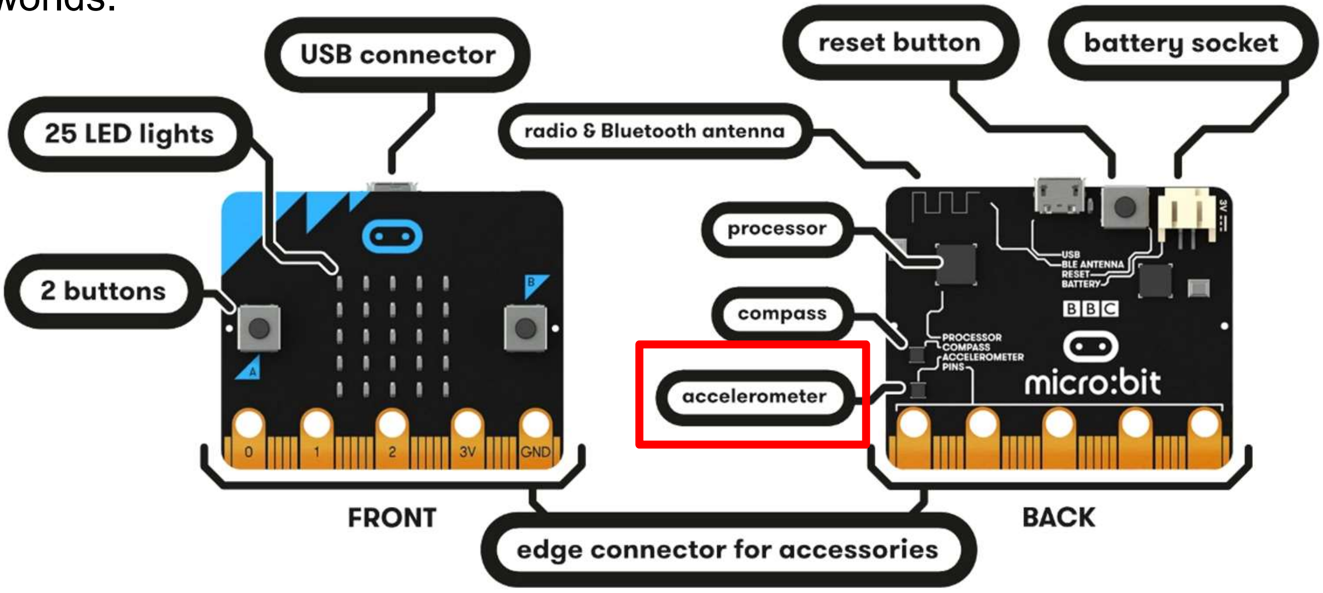
2. What will the panda sprite react when the hand-made joystick moves to the following directions? Match the sensing and reacting photos.



Creating a Maze Game with micro:bit

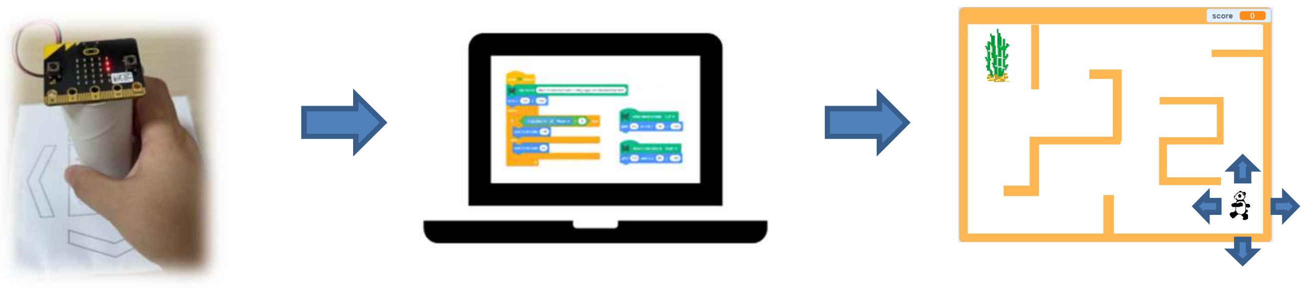
To Learn: micro:bit

Here we can see all the micro:bit's features, including an LED display, 2 buttons, and a motion sensor (accelerometer). You can connect it to Scratch and build creative projects that combine the magic of the digital and physical worlds.



In this unit, we will make use of micro:bit's feature – accelerometer, together with programming, to create a joystick to control the panda.

Knowledge builds up: Internet of Things (IoT) Sensing-Reasoning-Reacting



Sensing
Collect data from the sensors.

Reasoning
Judge and make decisions based on the data.

Reacting
Respond according to the result of reasoning.

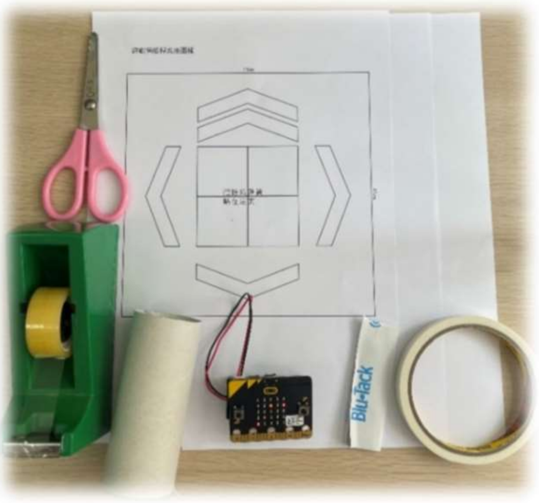
Creating a Maze Game with micro:bit

Pre-Lesson Task (Make the Joystick at Home)

You can try the followings to make a joystick model:

1. Prepare the following handicraft materials:

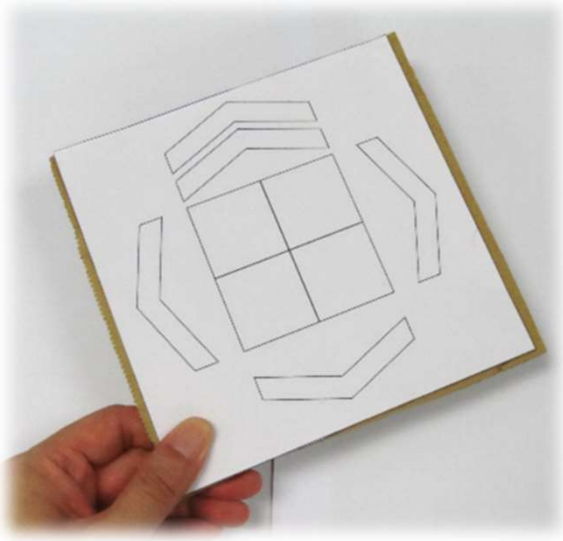
1



- Power-on micro:bit
- Toilet paper roll
- 2 pieces of A4 paper
- Paper pattern of base on Teacher Guide
- Cello tape
- Double-sided tape
- Scissors
- Blu-Tack

2. Print and cut out the Base Pattern of Joystick in next page (you can stick it to a cardboard with similar size, or simply stick on the table during the lesson at school).

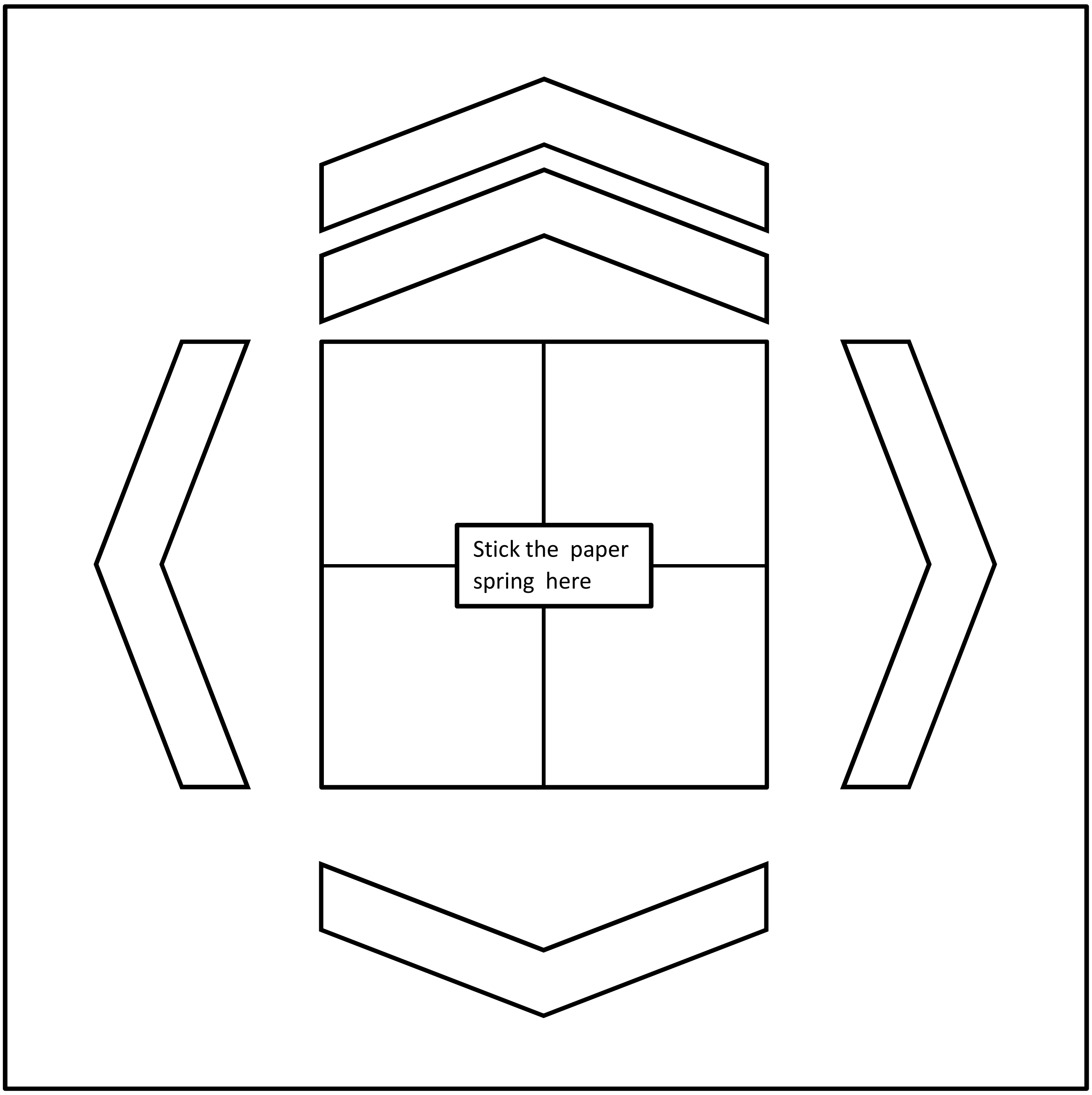
2



Creating a Maze Game with micro:bit

Pre-Lesson Task (Make the Joystick at Home)

Base Pattern of Joystick:



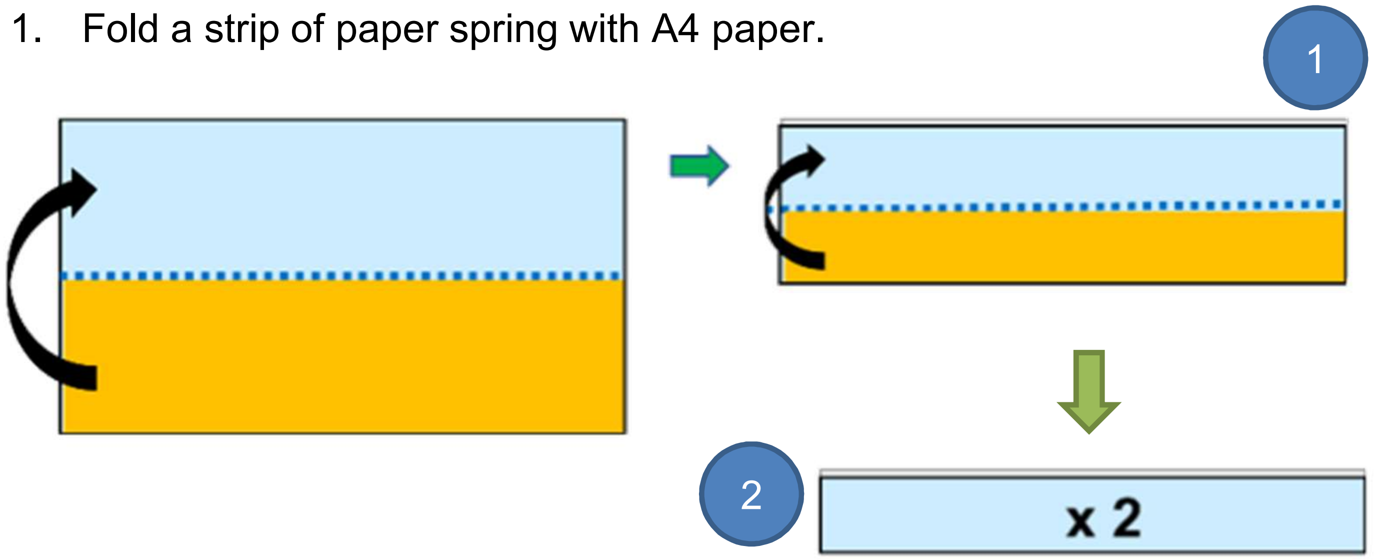
Teacher can remind students that this base pattern is not necessary. They can choose to cut out this page (the back is blank) or not.

Blank Page

Creating a Maze Game with micro:bit

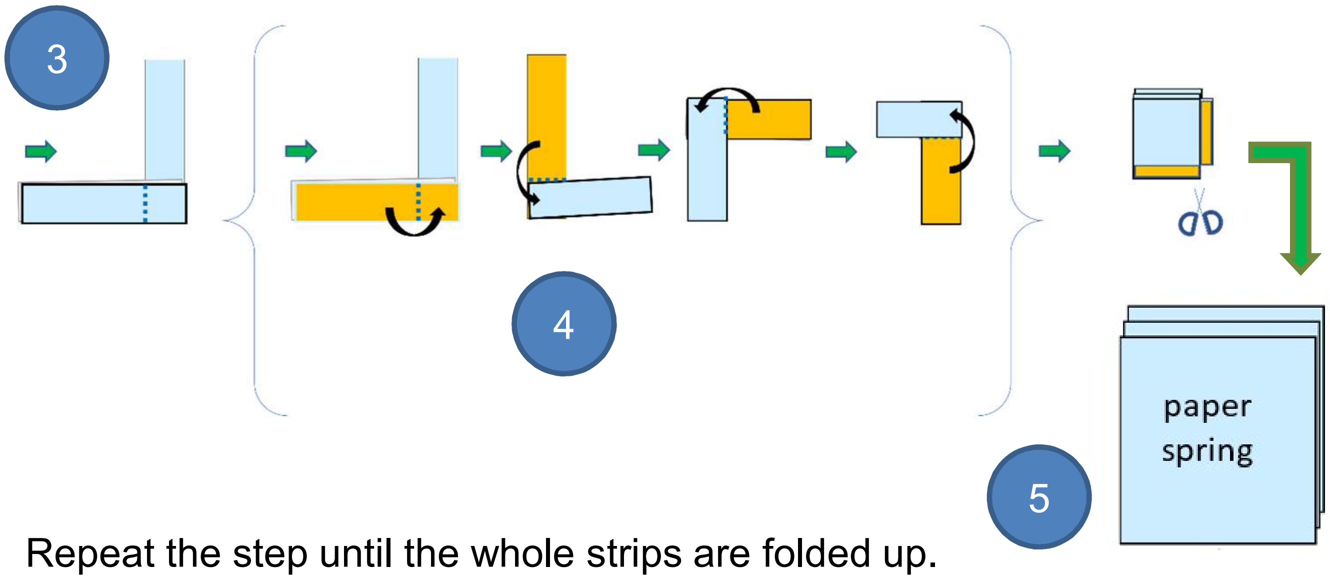
Pre-Lesson Task (Make the Joystick at Home)

1. Fold a strip of paper spring with A4 paper.



2. Fold two pieces of A4 paper in half vertically twice to form two thick strips.

3. Hold one strip horizontally with left hand and put another one to fix on the end of the horizontal strip with right hand.



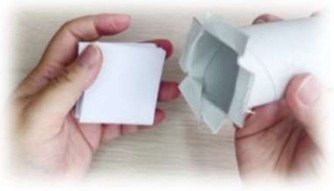
4. Repeat the step until the whole strips are folded up.

5. Cut off the excess parts and fix the ends with double-sided tape to form a paper spring.

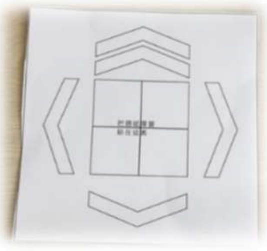
Creating a Maze Game with micro:bit

Pre-Lesson Task (Make the Joystick at Home)

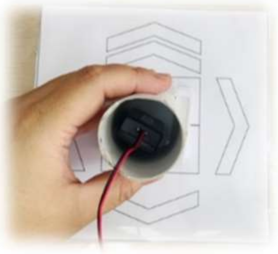
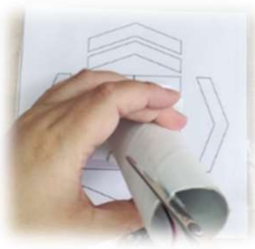
1. Cut four gaps about 1 cm long with similar distance at one end of the toilet paper tube with scissors; fold the gaps out and fix them with the paper spring with two-sided tape.



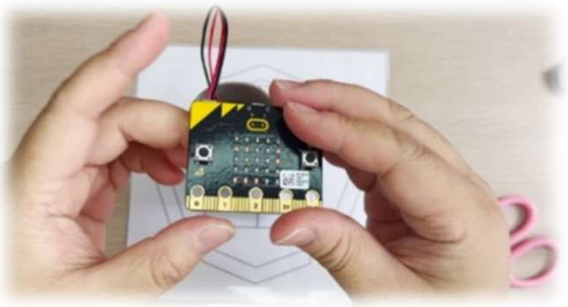
2. Fix another end of the paper spring to the centre of the paper base pattern with two-sided tape.



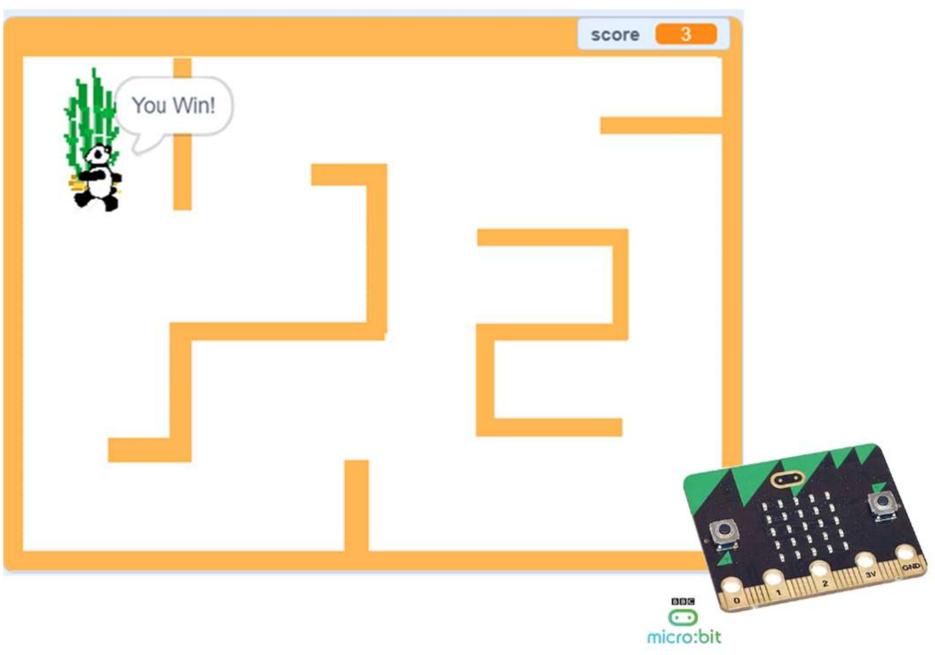
3. When you get the micro:bit from teacher (or you may complete last few steps at the class), put the battery box of micro:bit into the toilet paper tube, and cut a 4-cm long gap to hide the wire; and then stick the micro:bit to the top of the toilet paper tube with a Blu-Tack.

4. The joystick is made!



Creating a Maze Game with micro:bit



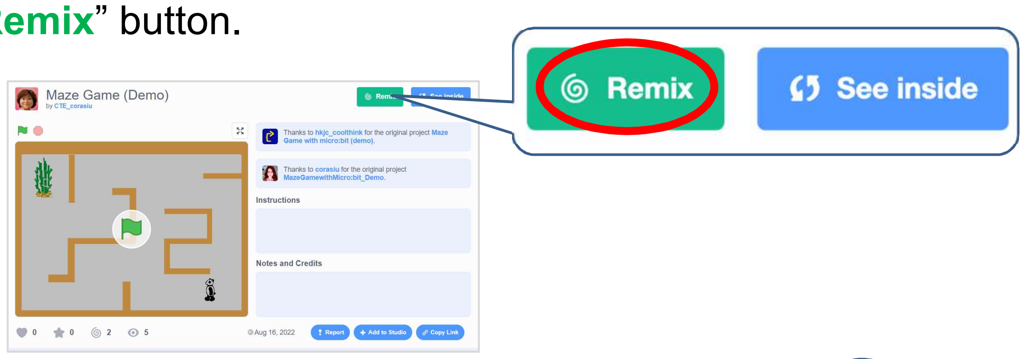
Start Here

1

1. Sign into your account at scratch.mit.edu.

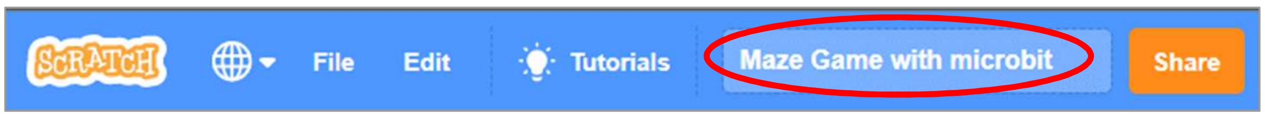
2. Open the Completed Maze Game in previous unit : https://scratch.mit.edu/projects/722154863 as the template for this unit and click on the "**Remix**" button.

2


Maze Game (Demo)
by CTE_corasiu
Remix
See inside
Thanks to hkjc_coolthink for the original project Maze Game with micro:bit (demo).
Thanks to corasiu for the original project MazeGamewithMicro:bit_Demo.
Instructions
Notes and Credits
Aug 16, 2022
Report
Add to Studio
Copy Link

3

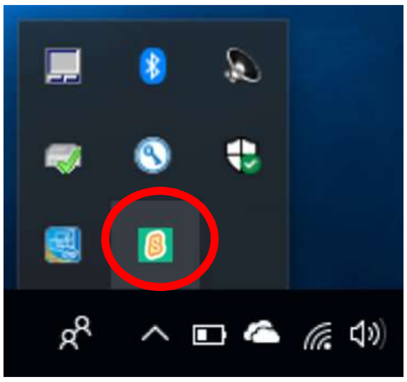
3. Rename the project as **Maze Game with microbit**.



Creating a Maze Game with micro:bit

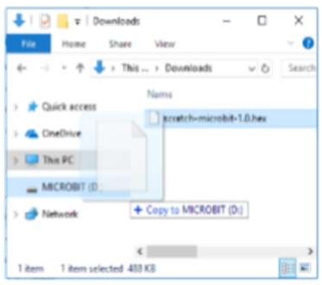
Prepare the Environment

1. To connect your micro:bit to Scratch, you need to download and install the Scratch Link software. You may find all the resources: https://scratch.mit.edu/microbit

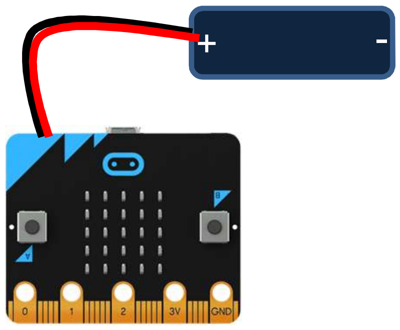


2. Start Scratch Link and make sure it is running. It should appear in your toolbar.

3. Download, drag and drop the Scratch micro:bit HEX file to the drive of micro:bit.



4. Connect the micro:bit with power using USB cable or battery box.

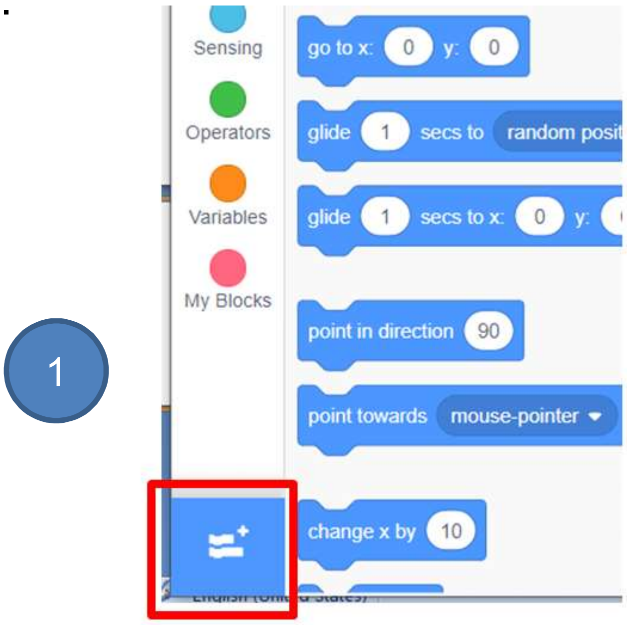


Creating a Maze Game with micro:bit

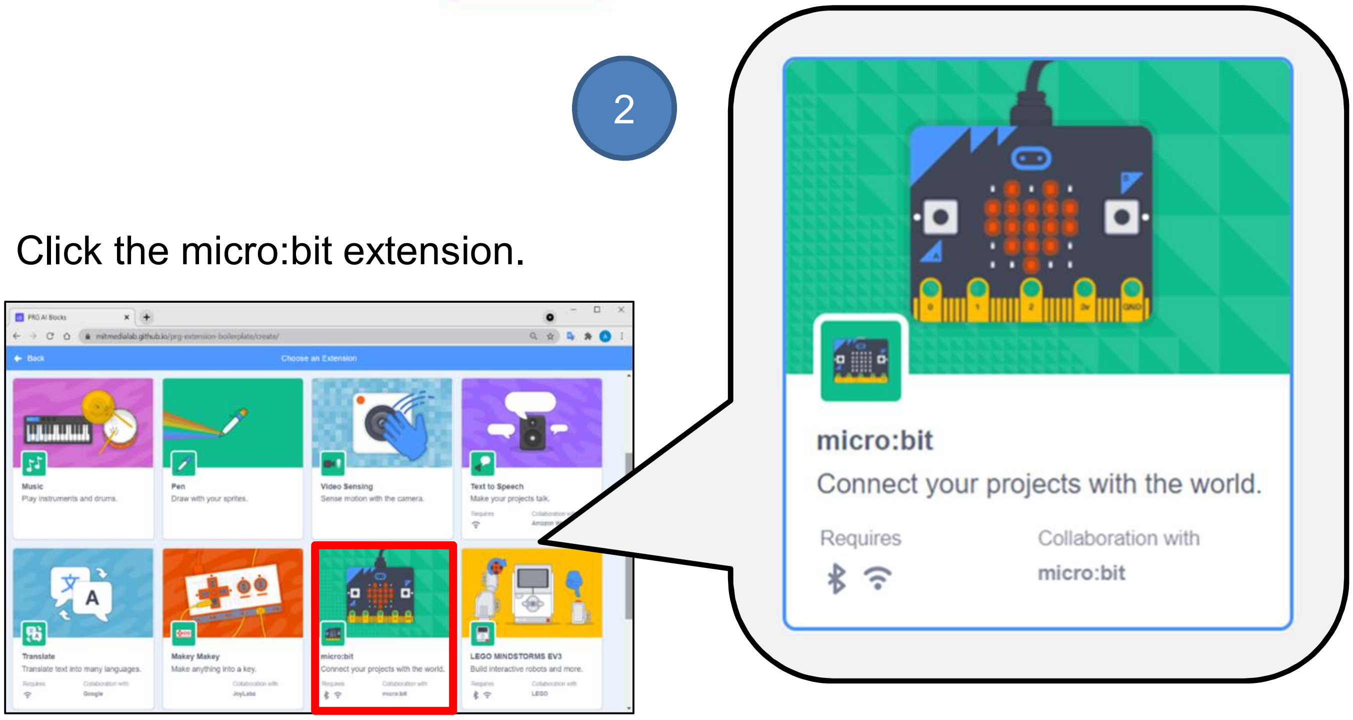
To Code: Add micro:bit Extension

Now go back to your Scratch project.

1. Click the button "**Add Extension**" on the bottom left corner of Scratch coding interface.



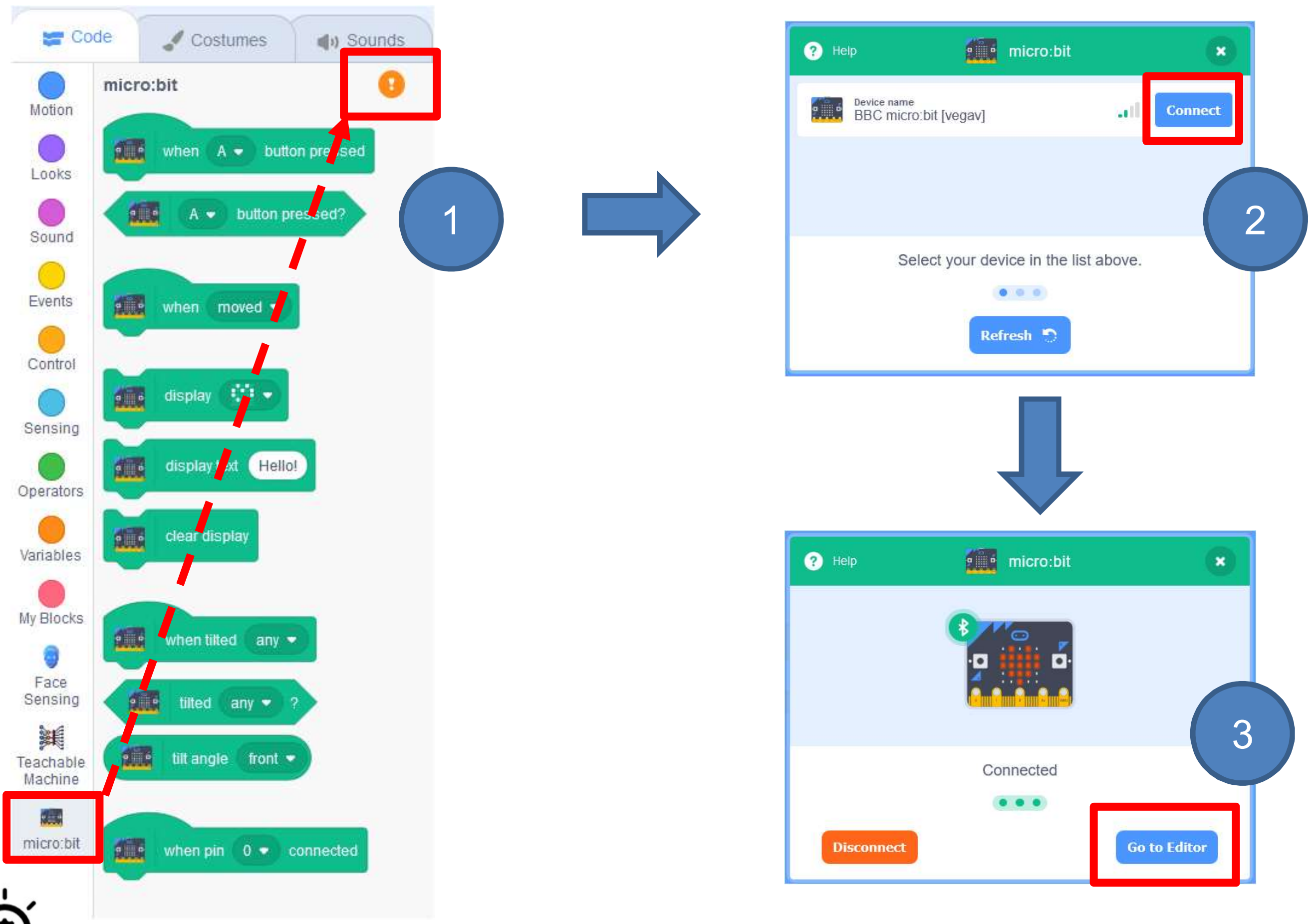
2. Click the micro:bit extension.


2
Choose an Extension
Music
Play instruments and drums.
Pen
Draw with your sprites.
Video Sensing
Sense motion with the camera.
Text to Speech
Make your projects talk.
Translate
Translate text into many languages.
Makey Makey
Make anything into a key.
micro:bit
Connect your projects with the world.
LEGO MINDSTORMS EV3
Build interactive robots and more.
micro:bit
Connect your projects with the world.
Requires
Collaboration with
micro:bit

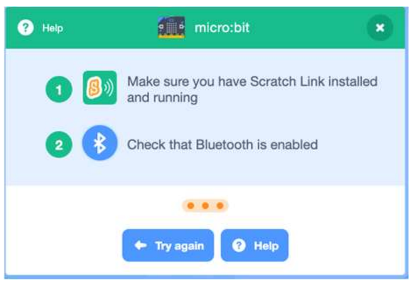
Creating a Maze Game with micro:bit

To Code: Connect micro:bit to Scratch

1. In the group "**micro:bit**" of Code tab page, click the icon ❗ (orange exclamation mark).
2. Click "**Connect**" button on the corresponding device option to connect the micro:bit.
3. Click **"Go to Editor"** button and return to the coding platform.



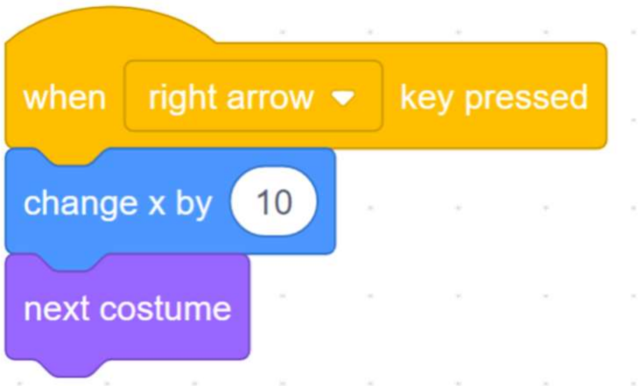
Knowledge builds up: Causal Reasoning
If you do not prepare the environment well or have something missing, e.g. Bluetooth turning off, low micro:bit battery, you may fail to connect micro:bit to Scratch as the screen shown.



1 Make sure you have Scratch Link installed and running
2 Check that Bluetooth is enabled

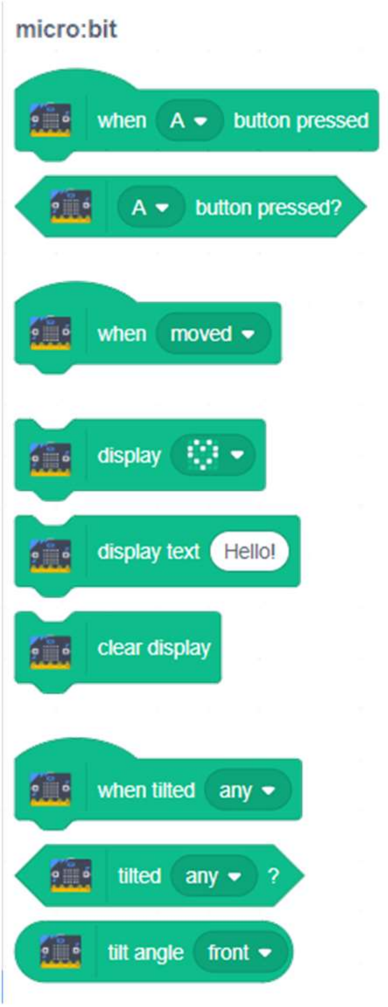
To Think and To Code: Control Panda with micro:bit

Compared with the program that we finished in the previous unit, think about which part of the program needs to be changed?



How can we update the code blocks so we control the Panda sprite with the control of joystick instead of the keyboard?

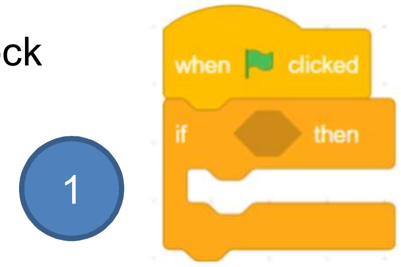
Hint: Look at the micro:bit Drawer.



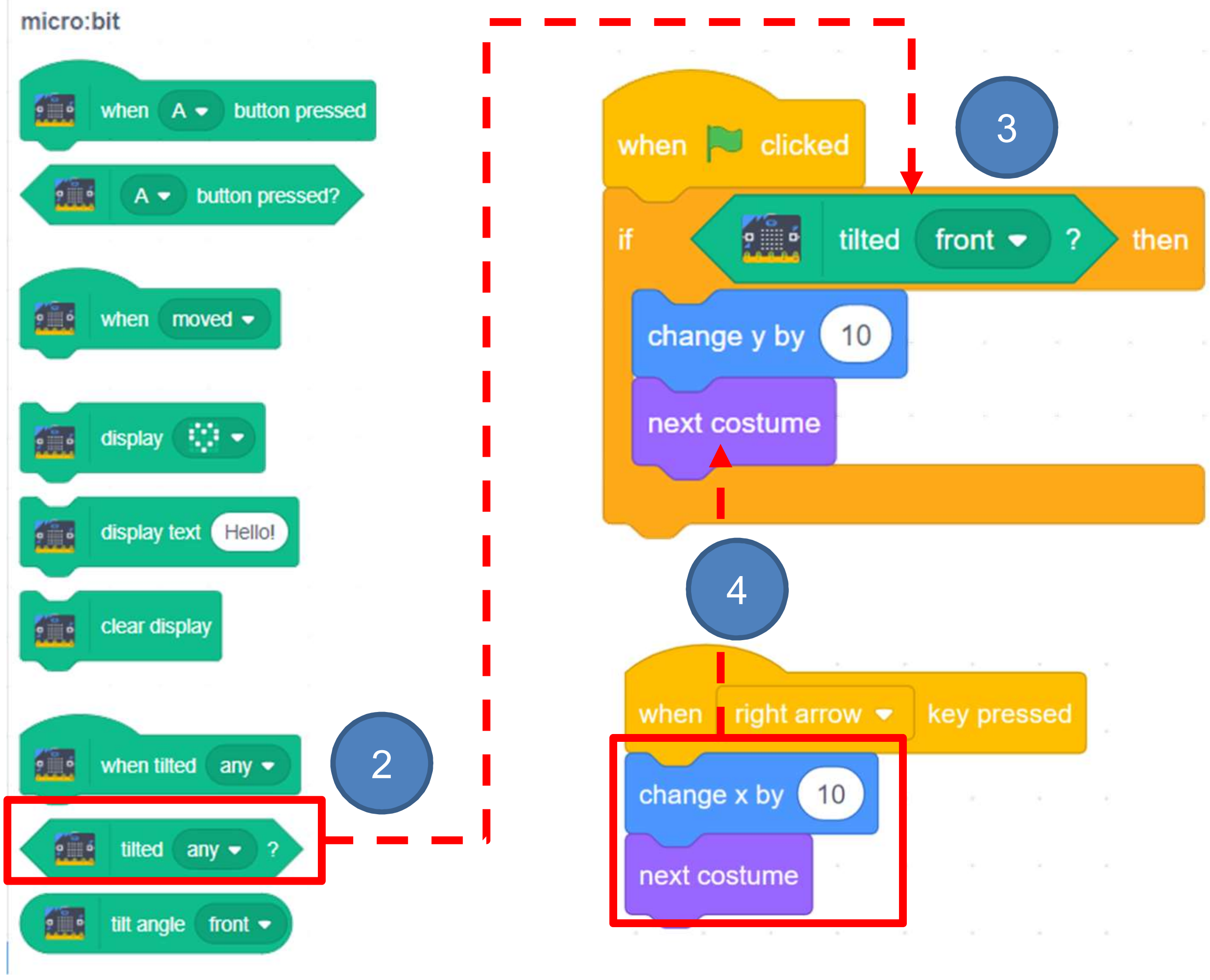
Creating a Maze Game with micro:bit

To Code: Control Panda with micro:bit

1. Drag a "**when green flag clicked**" & an **"if-then"** block and make them together.



2. From the new added micro:bit drawer, find the **"tilted <any>?"** block.
3. Change **"any"** to **"front"** and insert it into the **"if"** block.
4. Copy the code blocks of movement from **"when key pressed"** event.

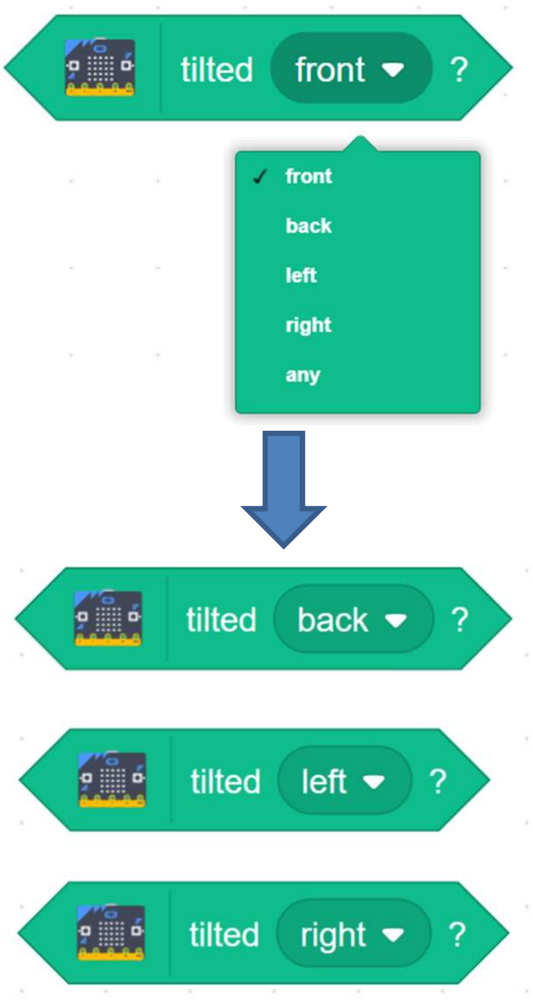


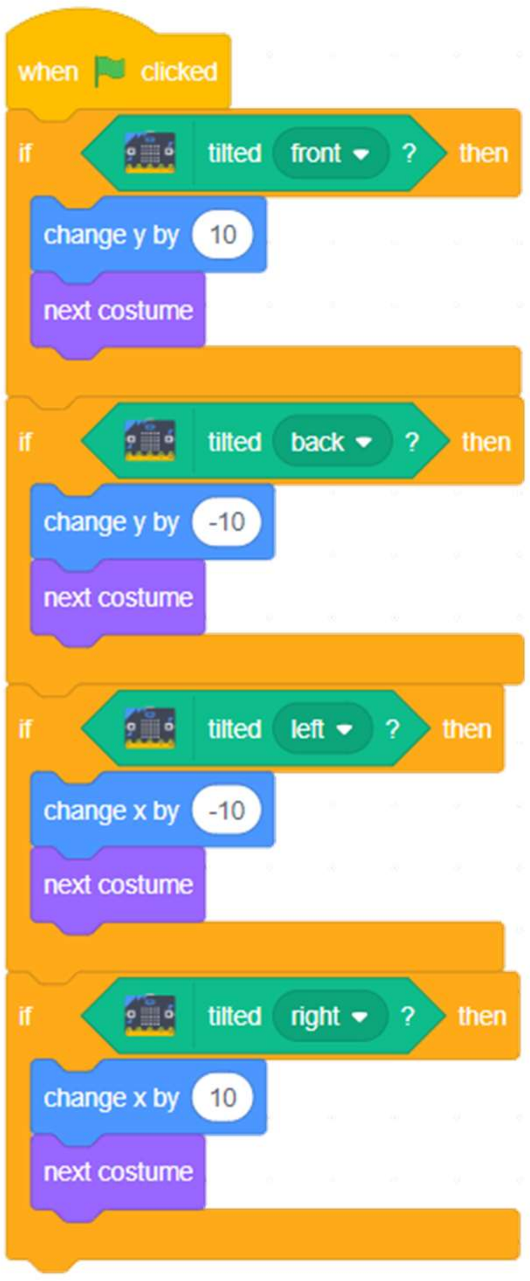
As "when tilted ___" can only be run once at a time, if we use "when tilted ___" to replace "when ____ pressed", then it will not have the game effect (keep the panda moving) of joystick.

Creating a Maze Game with micro:bit

To Code: Control Panda with micro:bit

Duplicate the **“titled front”** block, update and complete the rest of the other 3 directions:





```
when 🏴 clicked
if tilted front ? then
    change y by 10
    next costume
if tilted back ? then
    change y by -10
    next costume
if tilted left ? then
    change x by -10
    next costume
if tilted right ? then
    change x by 10
    next costume
```

Testing and Debugging

Can you control the sprite with micro:bit? Try more than one time? Which block do you miss?

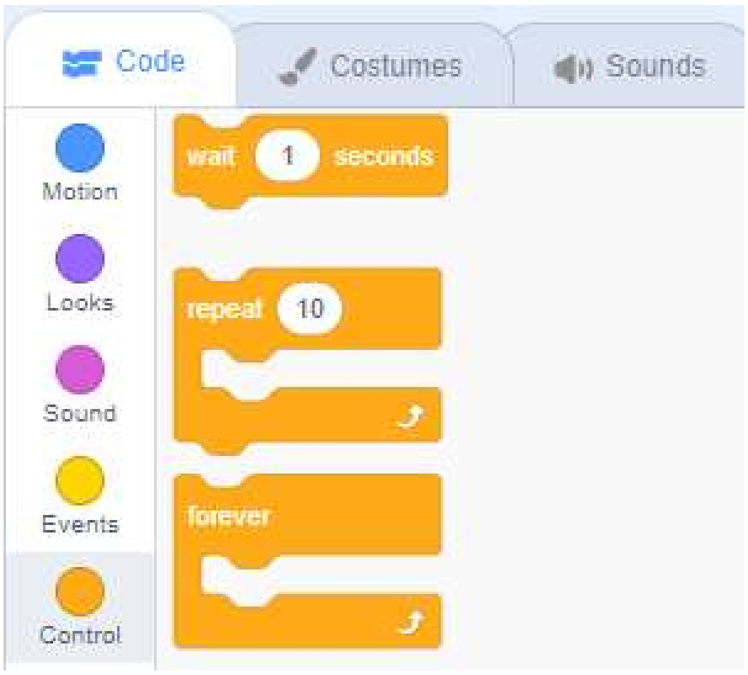
Control Panda with micro:bit ➡ ➡

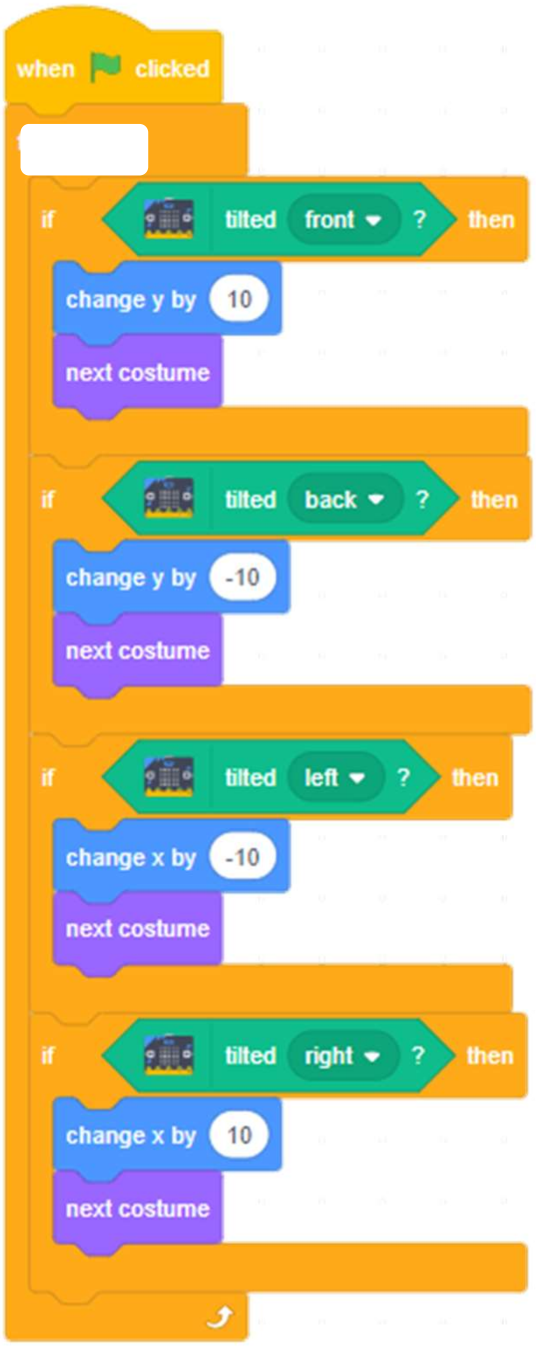
Creating a Maze Game with micro:bit

To Code: Iteration

Which block do you need to **solve** the problem?

Hint: Look for the **"Control"** drawer.




```
when [flag] clicked
[redacted]
  if <tilted front ?> then
    change y by 10
    next costume
  if <tilted back ?> then
    change y by -10
    next costume
  if <tilted left ?> then
    change x by -10
    next costume
  if <tilted right ?> then
    change x by 10
    next costume
```

Testing and Debugging

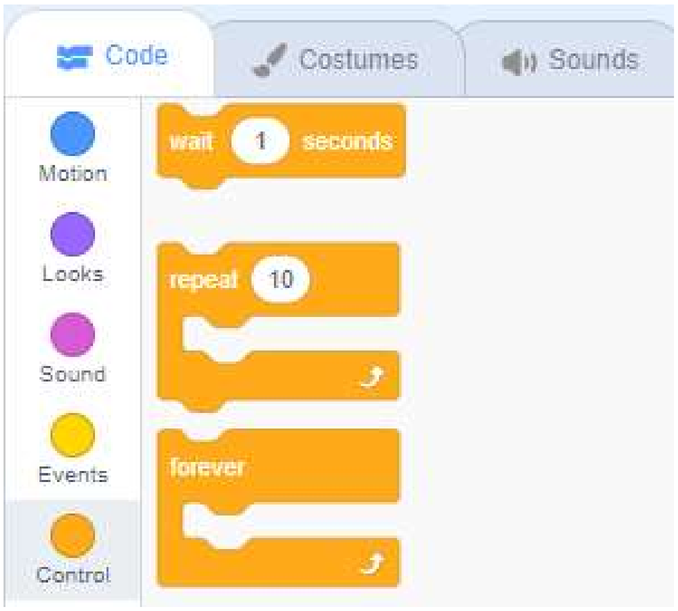
Try again! Is the joystick too sensitive?
What can you do to solve it?

Control Panda with micro:bit → Iteration →

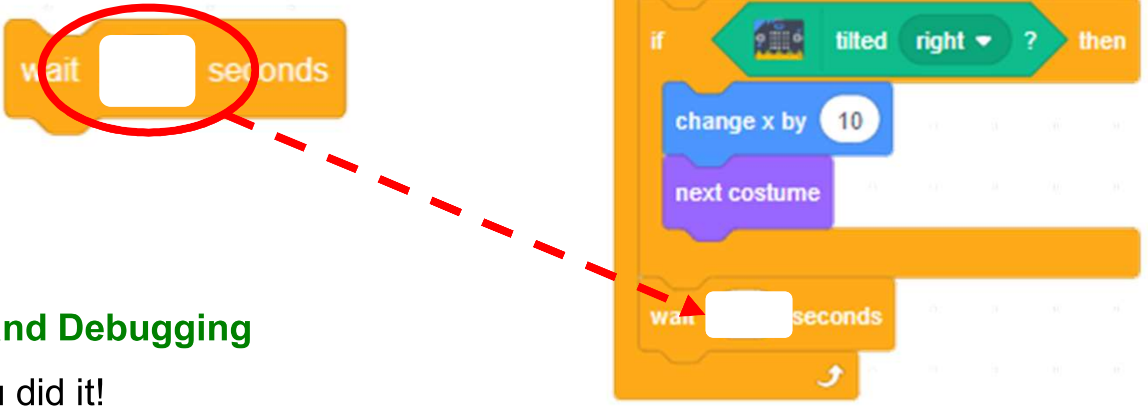
To Code: Add Wait Block

Is the joystick too sensitive? What can you do to solve it?

Add **"wait ___ seconds"** block see if it helps.



How long should we wait? You can input different value like 0.3, 1 second or 3 seconds until you find the appropriate time.



Testing and Debugging

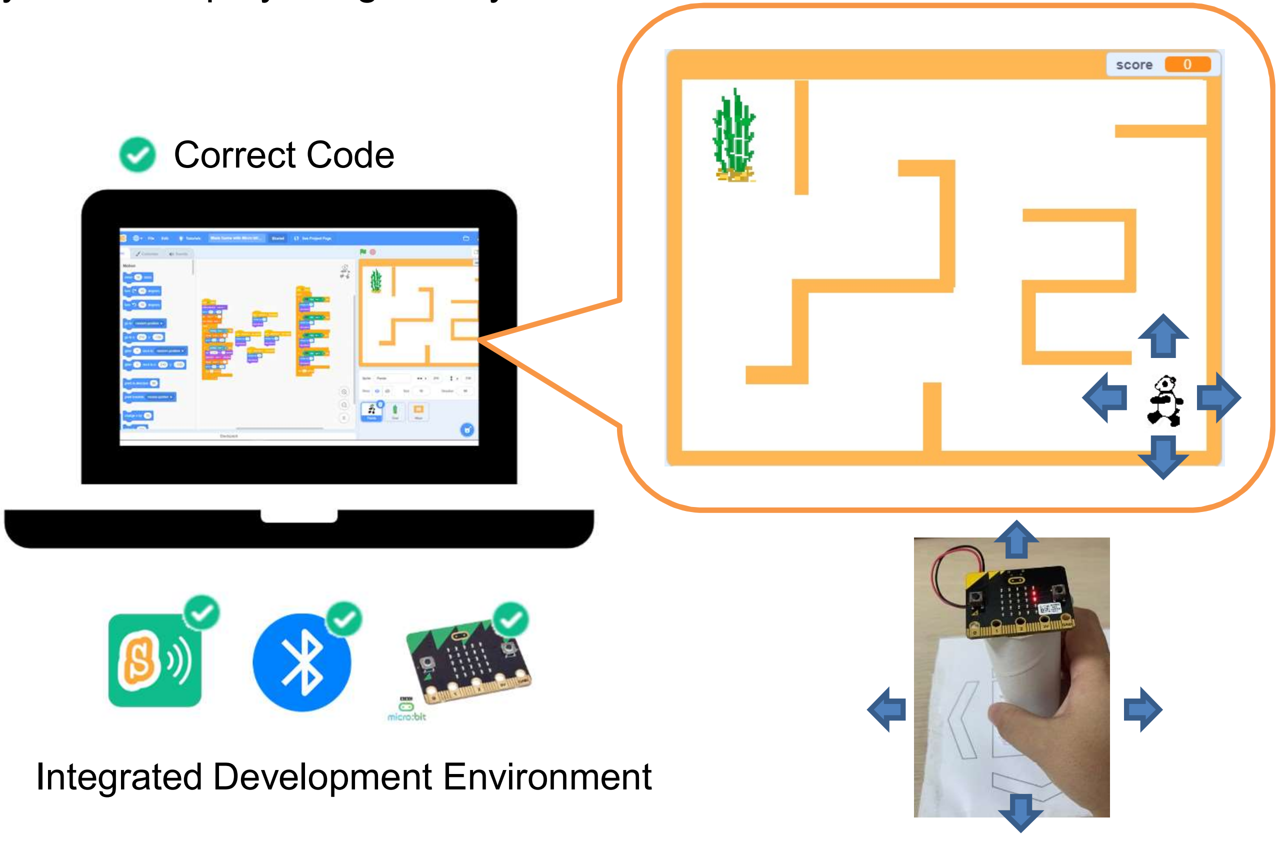
Yes! You did it!
Enjoy the maze game with micro:bit joystick!

Control Panda with micro:bit → Iteration → Add wait block

Creating a Maze Game with micro:bit

Knowledge builds up: Engineering Systems Thinking / Forming a system connected with physical objects

We should prepare the environment well and make everything connected or they will fail to play the game/system.

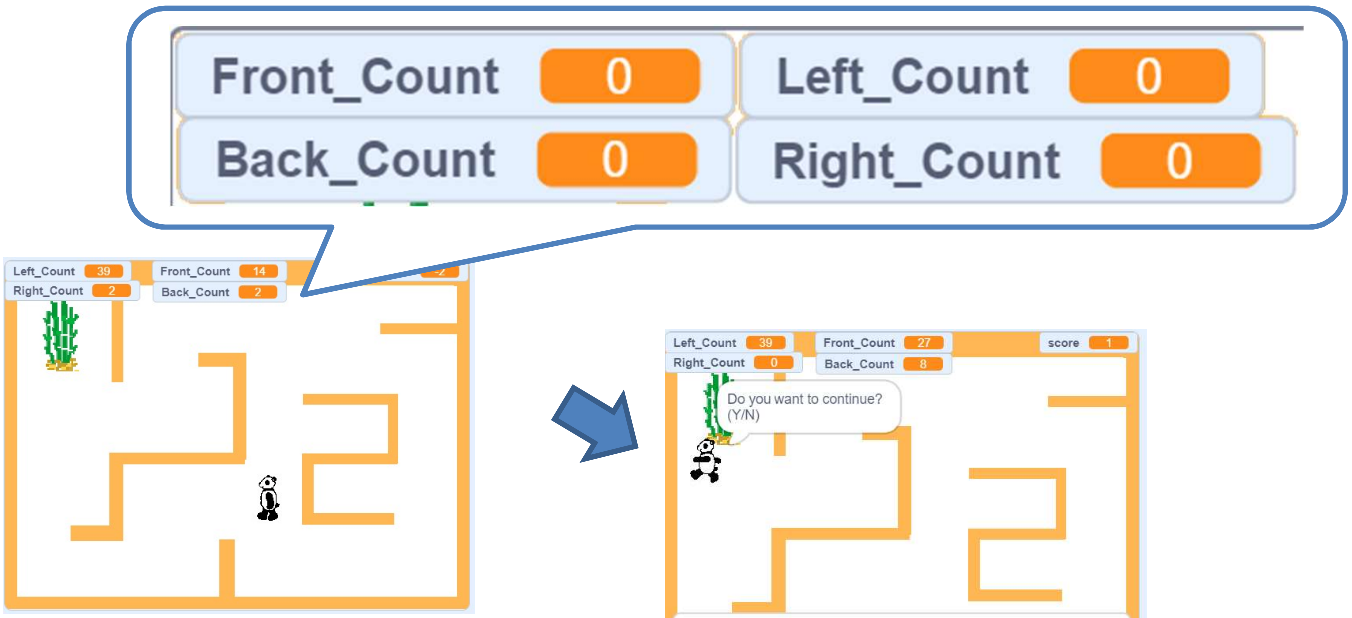


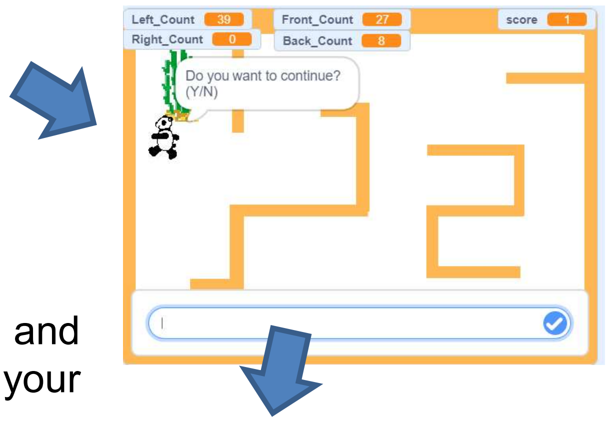
<u>Knowledge builds up: Engineering Systems Thinking /</u>
<u>Forming a system connected with physical objects</u>
It is about the thinking of putting interrelated parts together as a whole and enable the physical system work as it is intended.

Creating a Maze Game with micro:bit

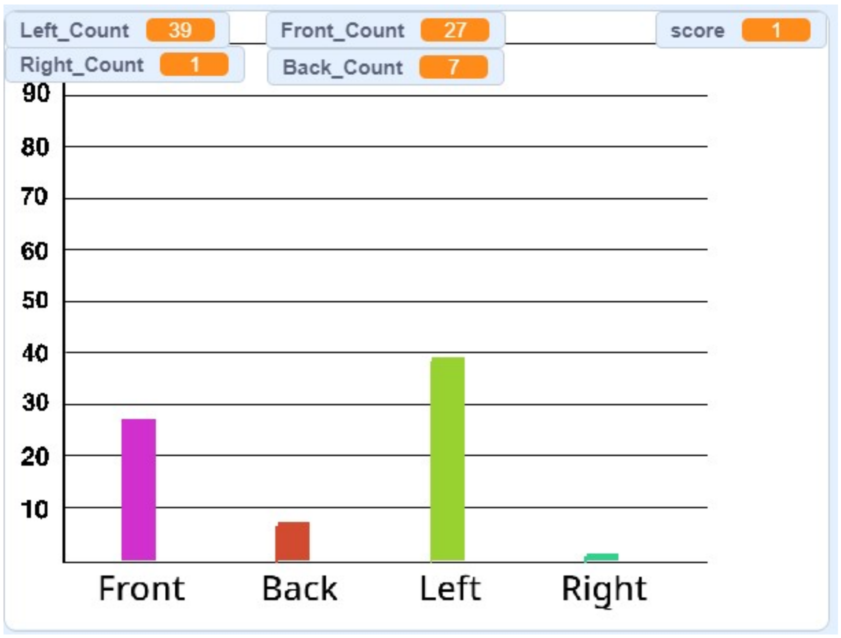
To Play: Maze Game with Data Analysis

Play another Maze Game with data analysis which counts your steps and shows you in bar graph. https://scratch.mit.edu/projects/734745981/.
Other than score, you will see four more variables to count the sprite moving front/back/left/right.





Finish the game by typing "N", and you will see a bar graph of your movement.

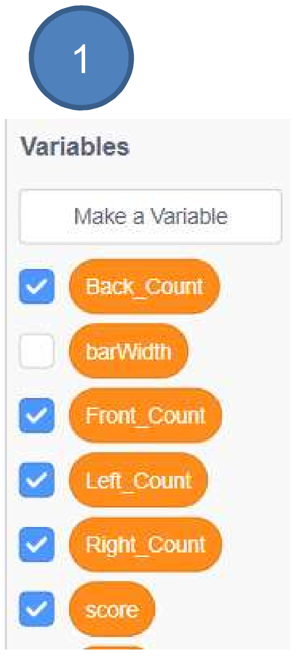


Creating a Maze Game with micro:bit

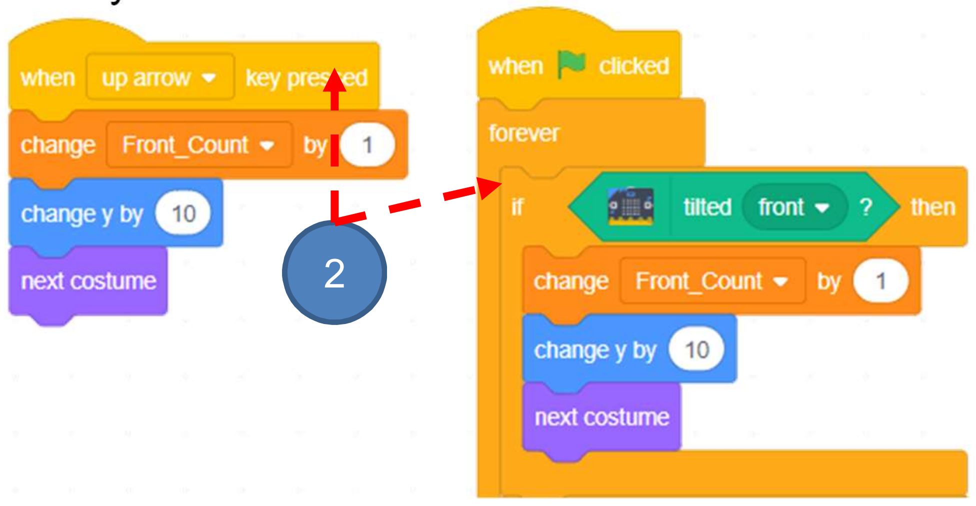
To Code: Code Comprehension

Click **“See inside”** to see the code.

1. There are four variables that counts your steps. It includes different variables to count your steps and shows data in the graph.
2. No matter you control with keyboard or micro:bit, it will count your moves.



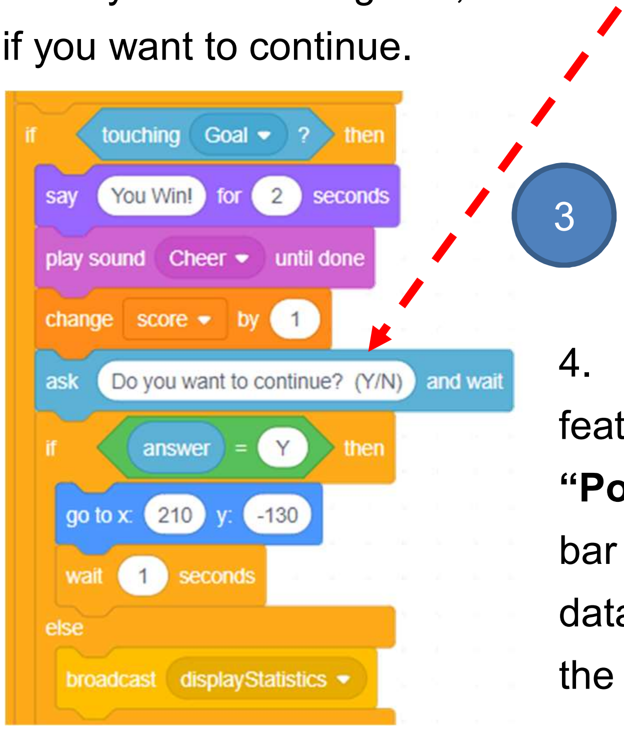
```
☑ Back_Count
☐ barWidth
☑ Front_Count
☑ Left_Count
☑ Right_Count
☑ score
```



```
when up arrow key pressed
change Front_Count by 1
change y by 10
next costume
```

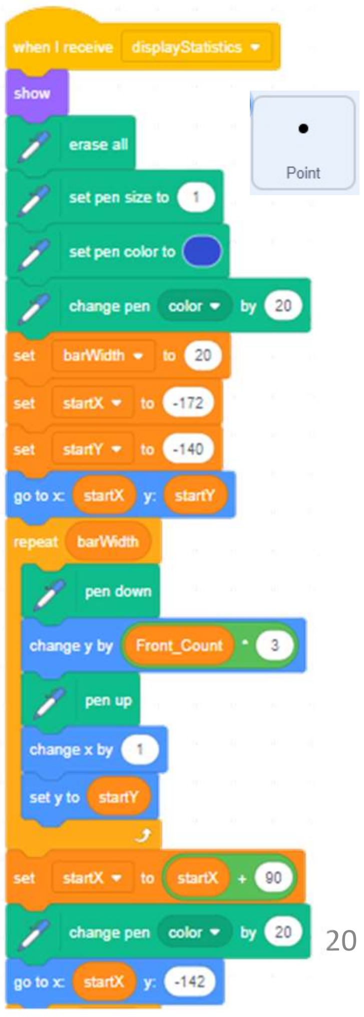
```
when 🏴 clicked
forever
    if tilted front ? then
        change Front_Count by 1
        change y by 10
        next costume
```

3. When you finish the game, it will **“ask”** you if you want to continue.



```
if touching Goal ? then
    say You Win! for 2 seconds
    play sound Cheer until done
    change score by 1
    ask Do you want to continue? (Y/N) and wait
    if answer = Y then
        go to x: 210 y: -130
        wait 1 seconds
    else
        broadcast displayStatistics
```

4. By using **“Pen”** feature, there is a **“Point”** Sprite to draw bar graph with those data it counts during the game.



```
when I receive displayStatistics
show
erase all
set pen size to 1
set pen color to ●
change pen color by 20
set barWidth to 20
set startX to -172
set startY to -140
go to x: startX y: startY
repeat barWidth
    pen down
    change y by Front_Count * 3
    pen up
    change x by 1
    set y to startY
set startX to startX + 90
change pen color by 20
go to x: startX y: -142
```

Creating a Maze Game with micro:bit

Conclusion

This unit is a taste of simplified IoT (Sensing – Reasoning – Reacting). With the joystick(micro:bit inside), we make use of the micro:bit build-in sensor (accelerometer) and write the code blocks to interact with the Scratch programming environment and form a system connected with physical objects.

Knowledge Build up – Internet Of Things (IoT)

Internet of Things (IoT): It is the network of physical things that embedded with sensors, apps and related technologies for the purpose of exchanging data with other devices over the internet. It usually use communication technologies such as Bluetooth and WiFi.

By using the Internet of Things, we can coordinate students' learning and activities during the school day.

Do you know any other real-life example with IoT?
e.g. Smart refrigerators and Light at home etc.

Creating a Maze Game with micro:bit

To Create

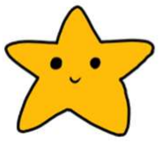
What do you want to create with the accelerometer of micro:bit you learnt today? Draw your idea in the box below:

Creating a Maze Game with micro:bit

To Reflect: Two Stars and a Wish Worksheet

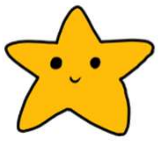
Name of Project: ____________ Name of Creator: ____________

Please write down two things that you like about this project.



1

"Two Stars and a Wish" is a reflection strategy designed for student feedback as peer- and self-assessment.
Teachers can guide students to give constructive feedback to their peers regarding their Scratch project - two positive (stars) and one hopeful (wish) reflection. Comments can be made on Scratch project's idea, features and aesthetic aspects etc.

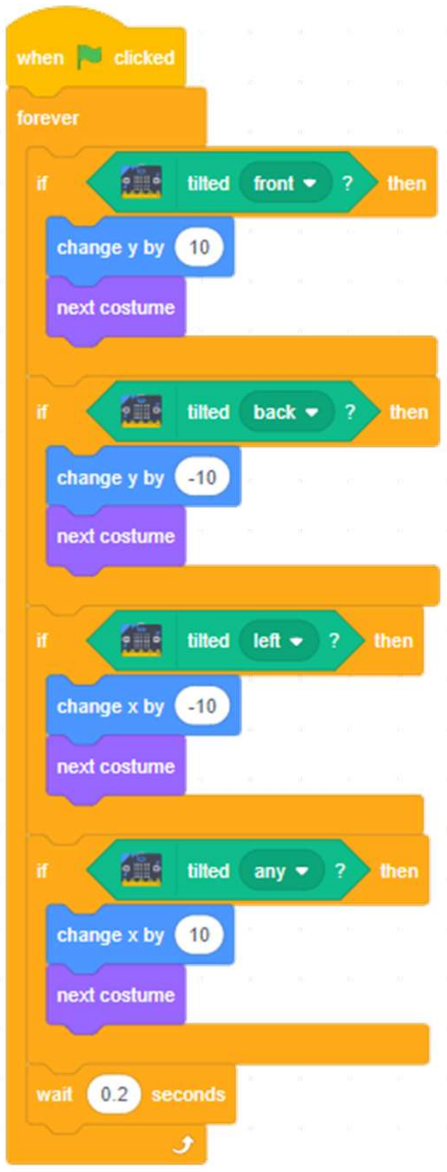


2

What is one thing you would like to add or change to make this project better?

Review Questions

1. A student tried controlling the sprite with micro:bit but failed to move to the correct direction. Read the following code blocks and try to debug.



```
when [flag] clicked
forever
    if <tilted front ?> then
        change y by 10
        next costume
    if <tilted back ?> then
        change y by -10
        next costume
    if <tilted left ?> then
        change x by -10
        next costume
    if <tilted any ?> then
        change x by 10
        next costume
    wait 0.2 seconds
```

A. Remove the forever block.

B. The wait seconds should be updated from “0.2” to “2”.

C. The forth if-then statement “tilted any?” should be replaced with “tilted right”.

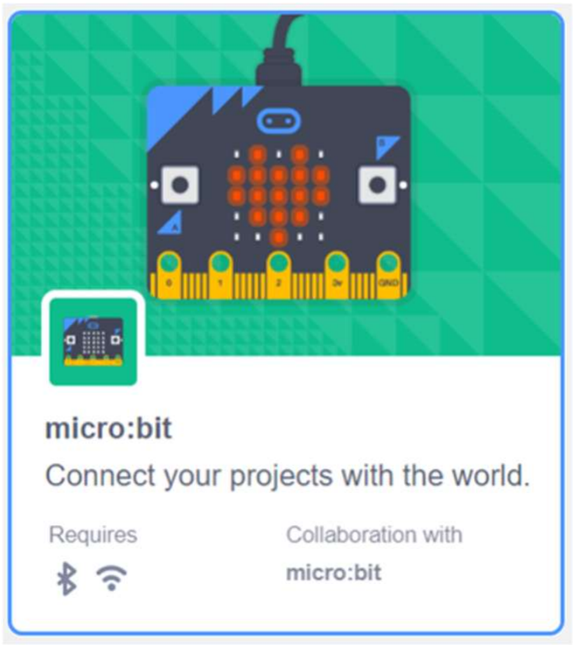
D. Use another micro:bit instead.

(Answer: C)

Creating a Maze Game with micro:bit

Revision on Key Features

micro:bit and accelerometer:

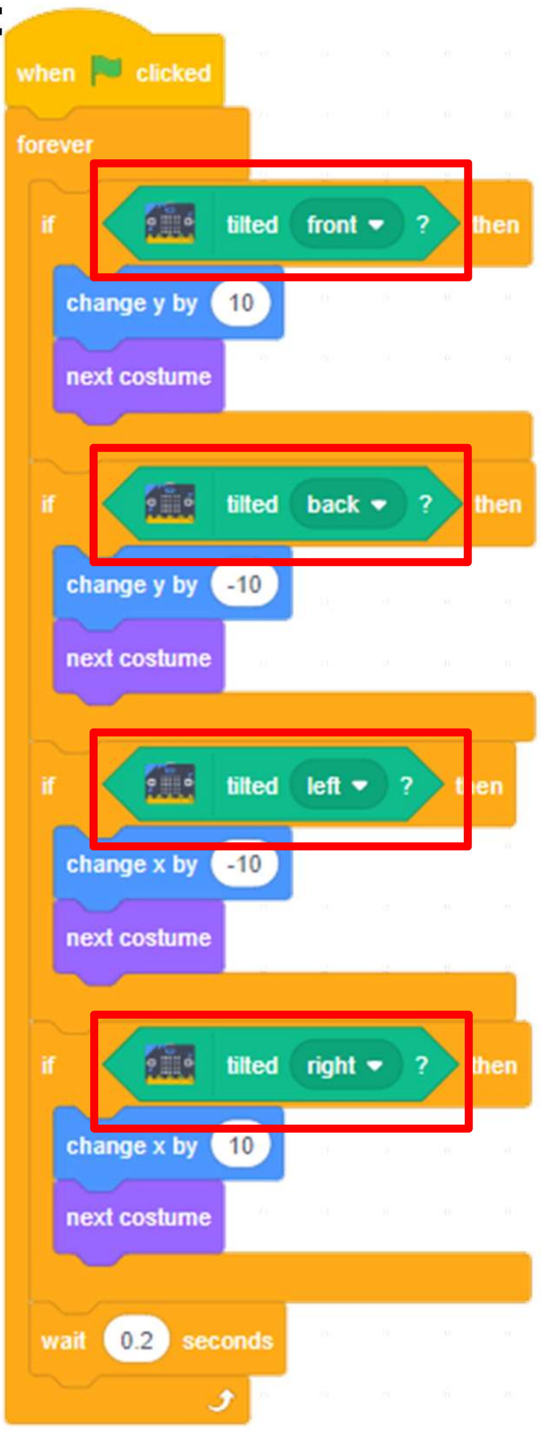


micro:bit

Connect your projects with the world.

Requires

Collaboration with micro:bit

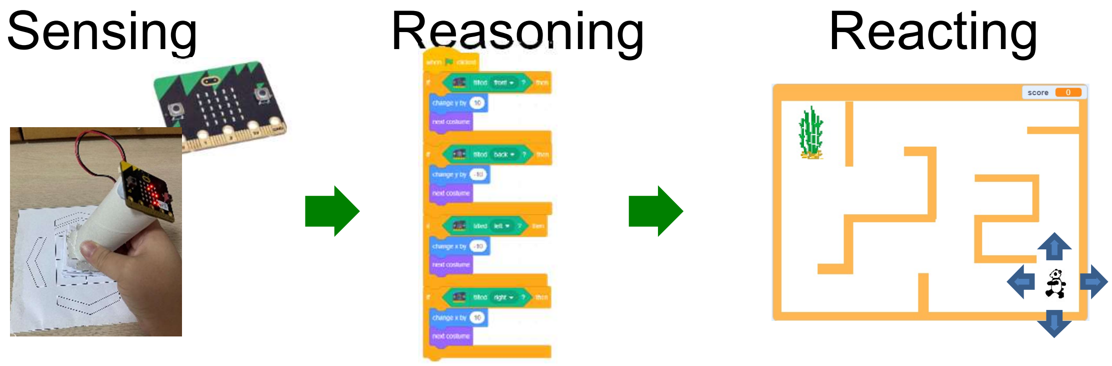


```
when ⚑ clicked
forever
    if tilted front ? then
        change y by 10
        next costume
    if tilted back ? then
        change y by -10
        next costume
    if tilted left ? then
        change x by -10
        next costume
    if tilted right ? then
        change x by 10
        next costume
    wait 0.2 seconds
```

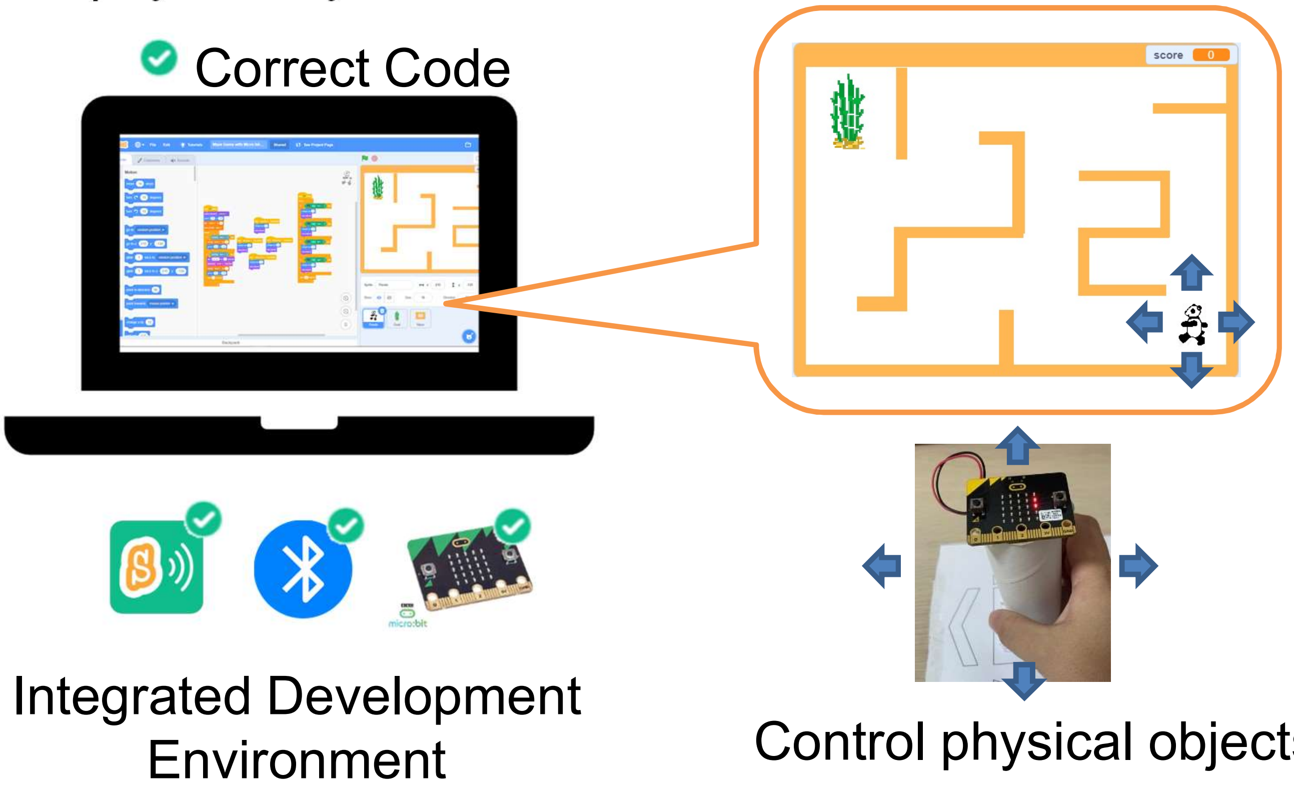
Creating a Maze Game with micro:bit

Revision on Key Concepts & Practices

Sensing: Collect data from the sensors.
Reasoning: Judge and make decisions based on the data.
Reacting: Respond according to the result of reasoning.

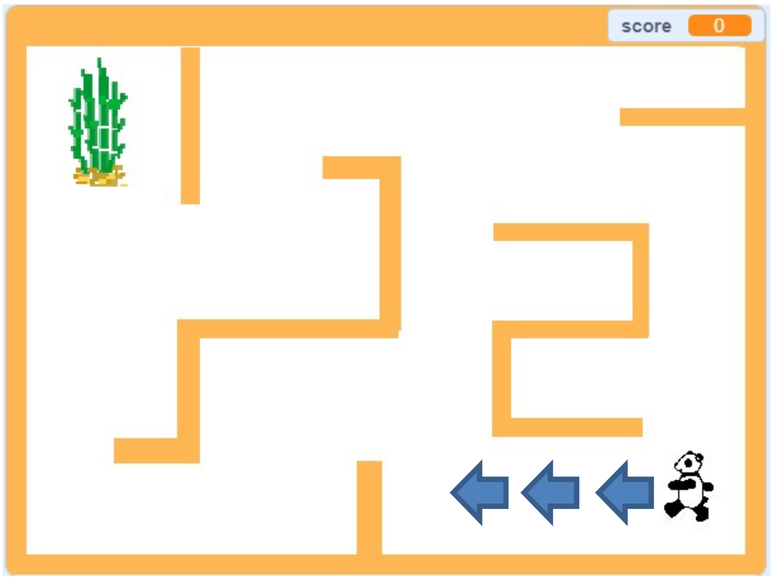


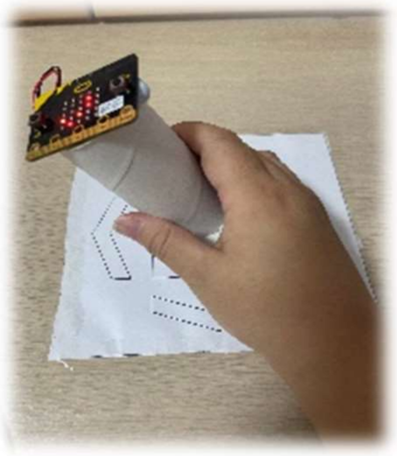
Engineering Systems Thinking / Forming a system connected with physical objects: It is about the thinking of putting interrelated parts together as a whole and enable the physical system work as it is intended.

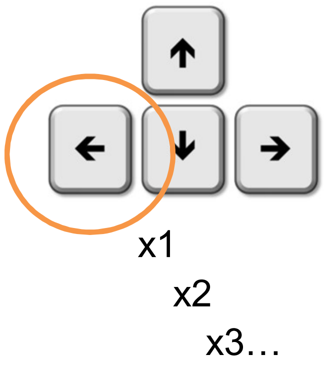


Revision on Key Concepts & Practices

Use the micro:bit's accelerometer (joystick) instead of the keyboard to control the Panda sprite.







Keep Moving

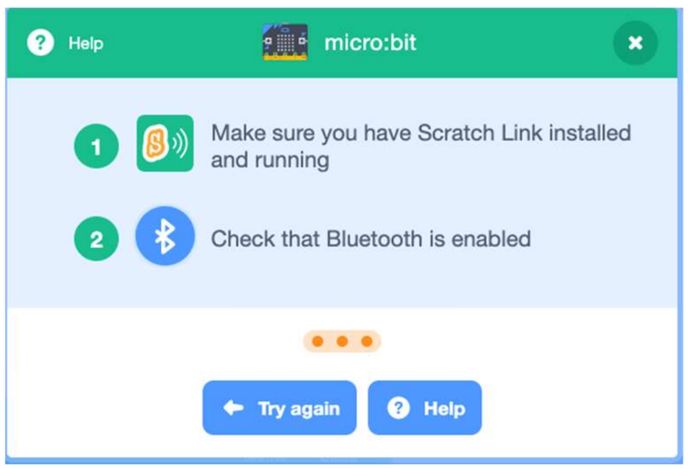
One Click, One Step

Creating a Maze Game with micro:bit

Revision on Key Concepts & Practices

Sequencing: The process of combining things in a specific order.

Casual Reasoning: The process of identifying causality: the cause and its effect.



1 Make sure you have Scratch Link installed and running

2 Check that Bluetooth is enabled

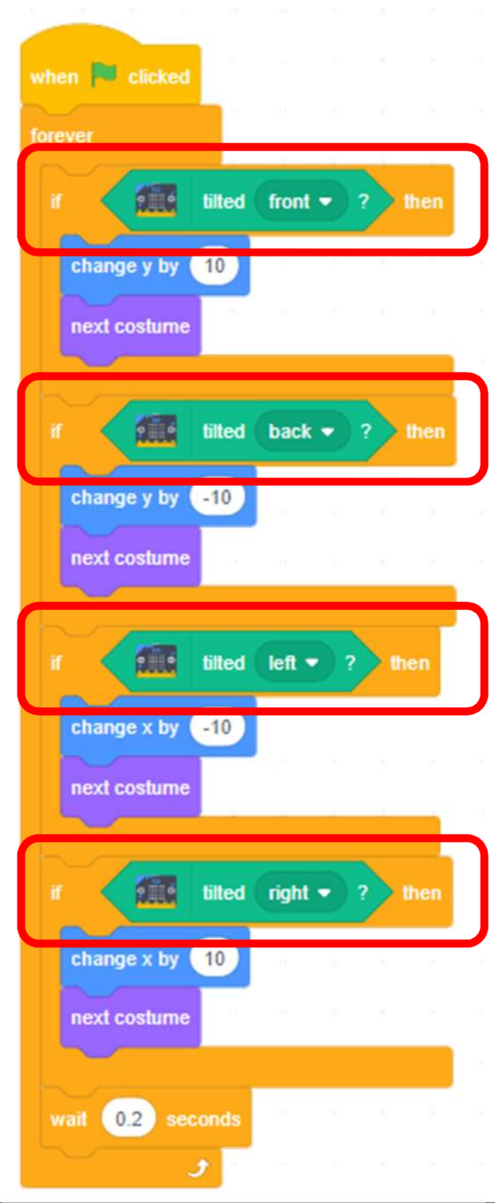
Error Screen

Checking Items

Creating a Maze Game with micro:bit

Revision on Key Concepts & Practices

Branching/Selection: We use conditional statements in programming to enable computers to make decisions. Conditionals always have an if part, which tells the program in the then part what to do when the condition is true.

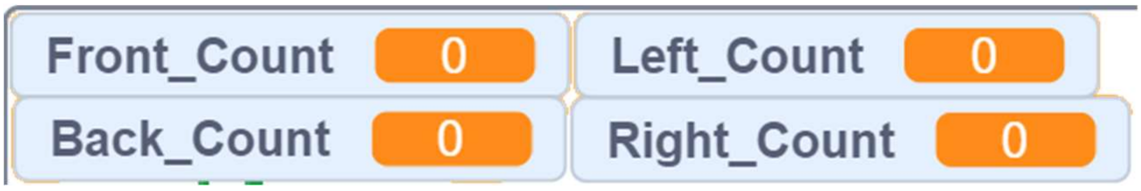


```
when ⚑ clicked
forever
    if tilted front ? then
        change y by 10
        next costume
    if tilted back ? then
        change y by -10
        next costume
    if tilted left ? then
        change x by -10
        next costume
    if tilted right ? then
        change x by 10
        next costume
    wait 0.2 seconds
```

Creating a Maze Game with micro:bit

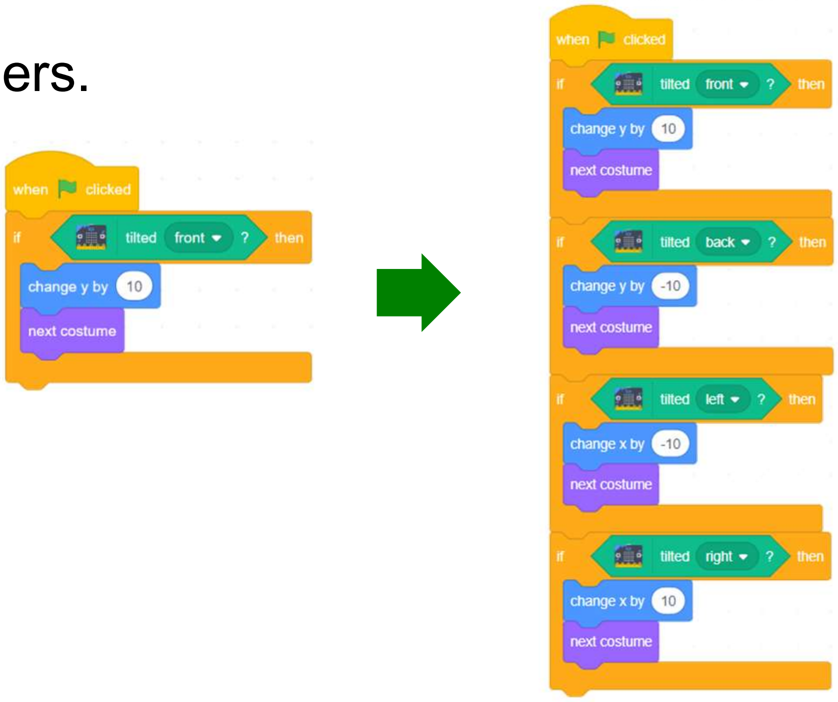
Revision on Key Concepts & Practices

Variables: In programming, variables are used to store values. It has a name, can only store one value at a time and be updated. For example, we use variables to count front/back/left/right key pressed events.



```
Front_Count  0     Left_Count   0
Back_Count   0     Right_Count  0
```

Reuse and Remix programs/codes: The reuse and remix of the works of other programmers are crucial in the online communities of Scratch. We can reuse and remix the codes of one micro:bit tilted event and use it for the others.



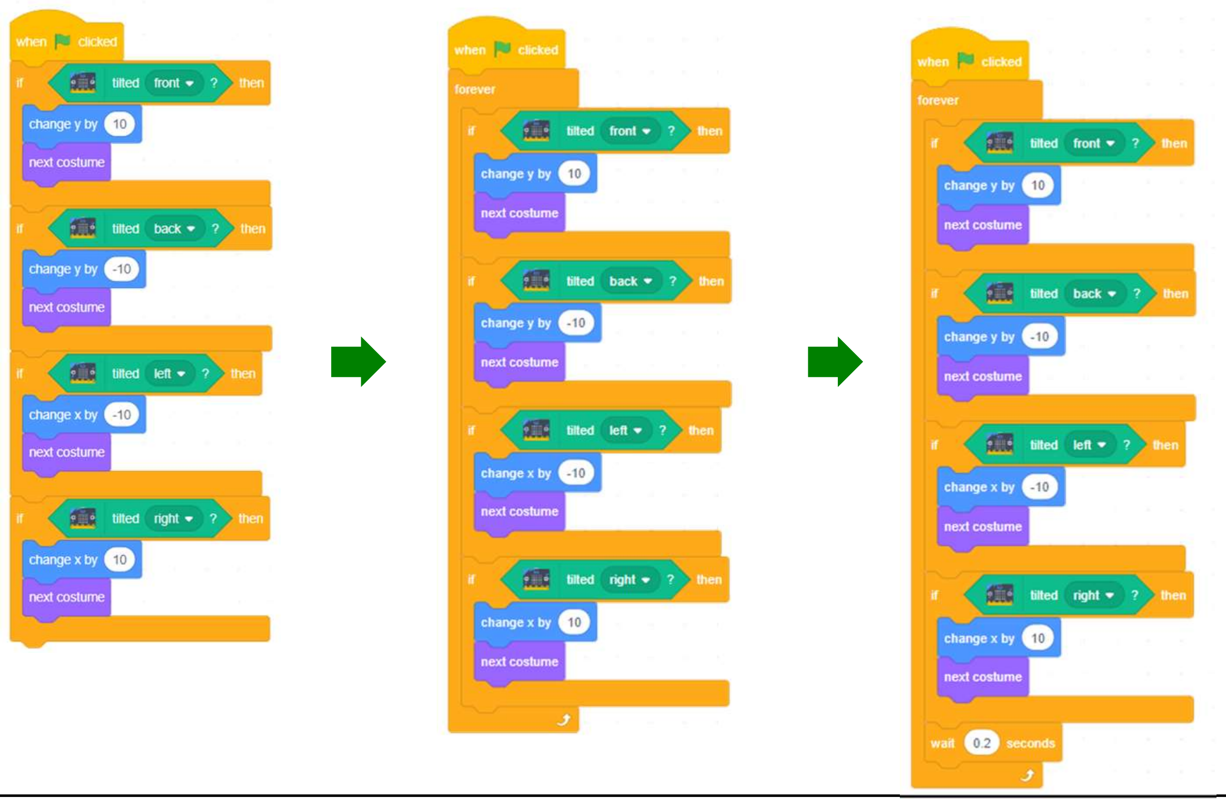
```
when [flag] clicked
if < [micro:bit] tilted (front ▾) ? > then
    change y by (10)
    next costume
```

```
when [flag] clicked
if < [micro:bit] tilted (front ▾) ? > then
    change y by (10)
    next costume
if < [micro:bit] tilted (back ▾) ? > then
    change y by (-10)
    next costume
if < [micro:bit] tilted (left ▾) ? > then
    change x by (-10)
    next costume
if < [micro:bit] tilted (right ▾) ? > then
    change x by (10)
    next costume
```

Creating a Maze Game with micro:bit

Revision on Key Concepts & Practices

Being incremental and iterative: to work out a sub-task as an iteration, try it out, then work out another sub-task in one more iteration until the whole programming task is completed.



```
when 🏁 clicked
if <tilted front ? > then
    change y by 10
    next costume
if <tilted back ? > then
    change y by -10
    next costume
if <tilted left ? > then
    change x by -10
    next costume
if <tilted right ? > then
    change x by 10
    next costume
```

```
when 🏁 clicked
forever
    if <tilted front ? > then
        change y by 10
        next costume
    if <tilted back ? > then
        change y by -10
        next costume
    if <tilted left ? > then
        change x by -10
        next costume
    if <tilted right ? > then
        change x by 10
        next costume
```

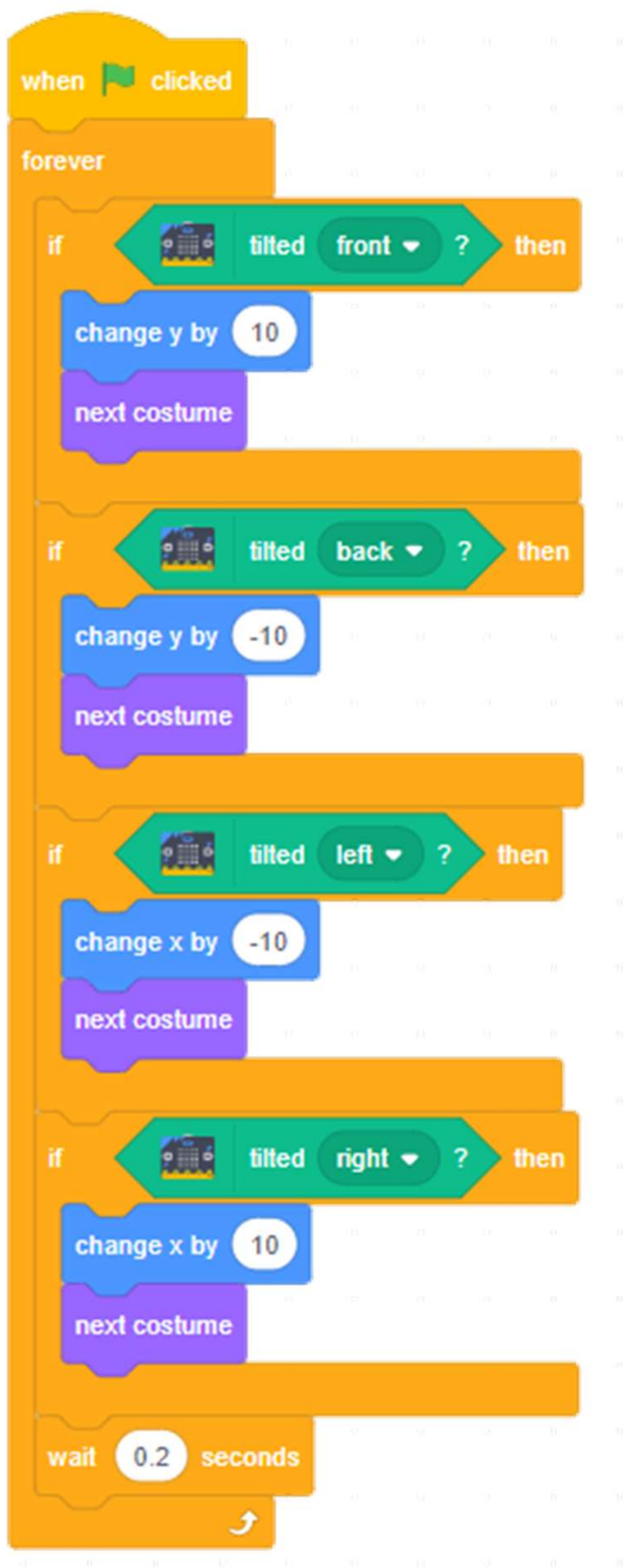
```
when 🏁 clicked
forever
    if <tilted front ? > then
        change y by 10
        next costume
    if <tilted back ? > then
        change y by -10
        next costume
    if <tilted left ? > then
        change x by -10
        next costume
    if <tilted right ? > then
        change x by 10
        next costume
    wait 0.2 seconds
```

Testing and Debugging: Testing a computer program is the process of checking if it can produce outcomes as designed. Debugging a computer program is the process of finding out ways to revise the program so that the bugs can be removed.

Control Panda with micro:bit → Iteration → Add Wait Block

Creating a Maze Game with micro:bit

Program Codes



```
when [flag] clicked
forever
    if <tilted front ?> then
        change y by 10
        next costume
    if <tilted back ?> then
        change y by -10
        next costume
    if <tilted left ?> then
        change x by -10
        next costume
    if <tilted right ?> then
        change x by 10
        next costume
    wait 0.2 seconds
```

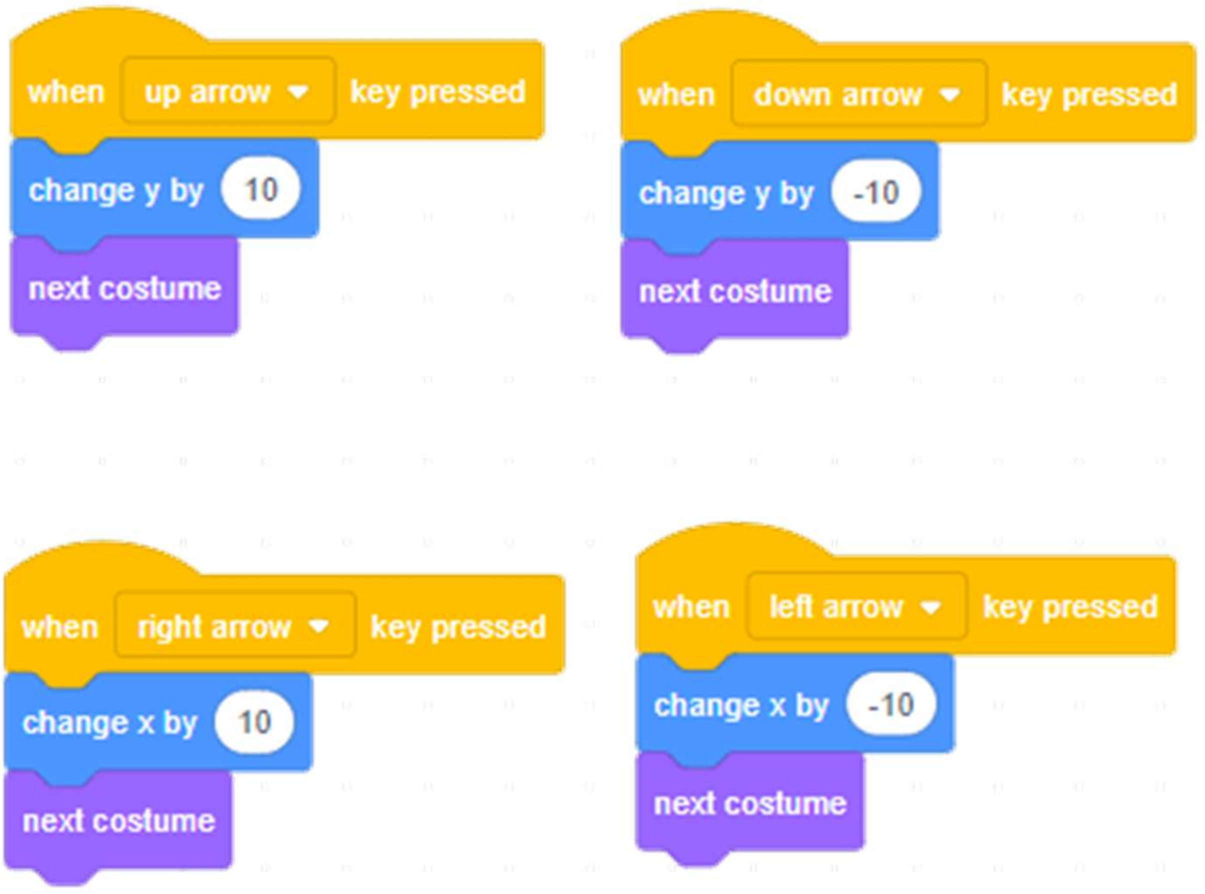
Sprite
(Panda)

Creating a Maze Game with micro:bit

Program Codes

Given in the template

Sprite (Panda)

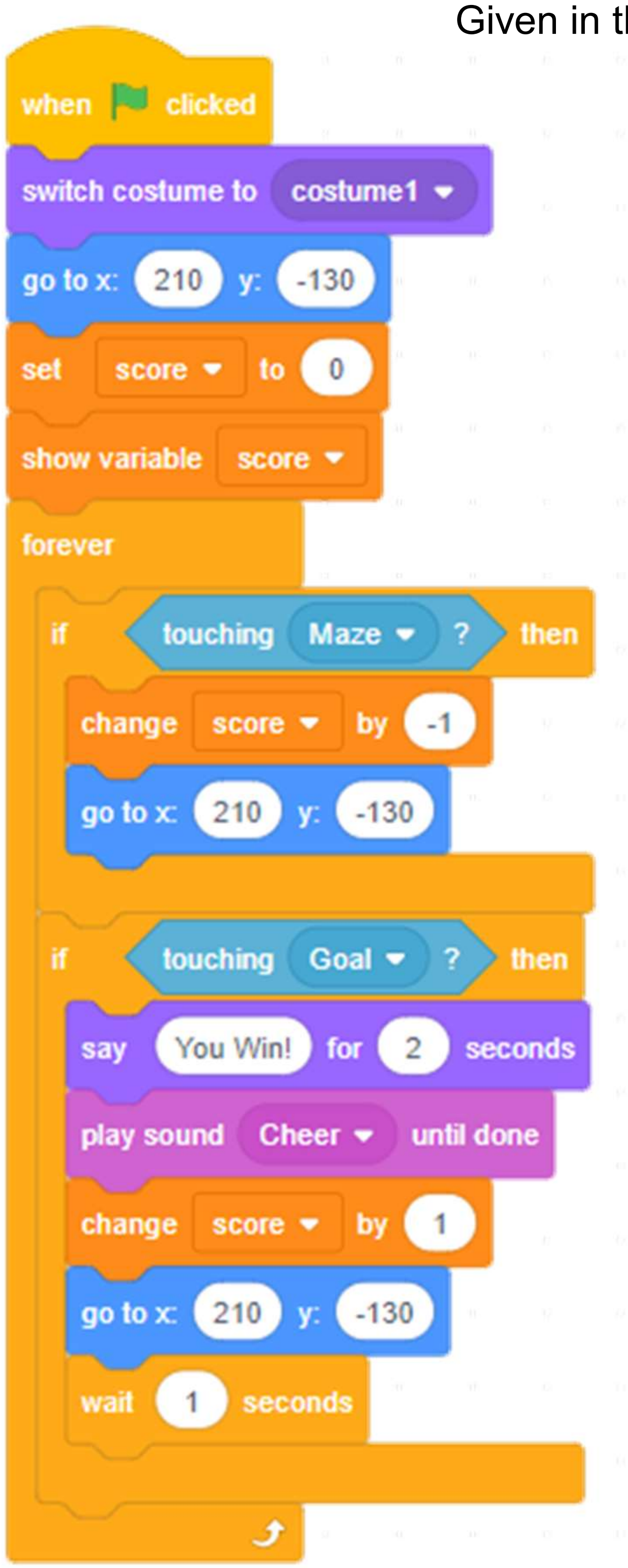


```
when [up arrow ▾] key pressed
change y by (10)
next costume

when [down arrow ▾] key pressed
change y by (-10)
next costume

when [right arrow ▾] key pressed
change x by (10)
next costume

when [left arrow ▾] key pressed
change x by (-10)
next costume
```

Creating a Maze Game with micro:bit

Program Codes

Given in the template



```
when [flag] clicked
switch costume to (costume1 ▾)
go to x: (210) y: (-130)
set [score ▾] to (0)
show variable [score ▾]
forever
    if <touching (Maze ▾) ?> then
        change [score ▾] by (-1)
        go to x: (210) y: (-130)
    if <touching (Goal ▾) ?> then
        say (You Win!) for (2) seconds
        play sound (Cheer ▾) until done
        change [score ▾] by (1)
        go to x: (210) y: (-130)
        wait (1) seconds
```

Sprite (Panda)

Unit 7: Drawing Shapes in Scratch Teacher Guide

Content

Unit 7: Drawing Shapes in Scratch
Teaching Plan

Prior Knowledge

Students should have experience in programming with Scratch. They should know how to add extension, use repeat block and move sprites with steps. They should also know some properties of a square such as a square contains four equal sides and there are four right-angles.

Learning Objectives

1. Create a Scratch project that uses the Pen feature to draw shapes;
2. Apply iteration for drawing squares and use these squares to create more complex and interesting shapes;
3. Develop computational thinking concepts and practices of abstraction and modularization through creating procedures to draw shapes;
4. Inspire students to be creative in drawing shapes using coding skills.

Learning Elements

Computational Thinking Concepts and Practices:

Key Learning Elements	Items
Abstraction	Express the algorithm, Modularization
Algorithm	Problem Solving Procedures: Programming Basic Programming Constructs: Sequence, Iteration Coding Concept and Practices: Design, Reuse and Remix programs / codes, Testing and Debugging

Coding Skills:

1. Make use of the Pen feature in Scratch;
2. Create and use “My Block” (procedure);
3. Reuse and remix programs / codes from drawing a line to a square;
4. Apply testing and debugging in completing each task of this unit.

Others (including Attitude):

1. Develop interest in programming;
2. Show perseverance and positivity in testing and debugging;
3. Inspire students to be creative and innovative to draw shapes.

Lesson Plan: This unit consists of 3 lessons of 35 minutes.

Lesson 1

Time	**Activity**
5 mins	**To Play** 1. Open the Scratch Project, Drawing Shapes in Scratch: https://scratch.mit.edu/projects/737338011/. 2. Ask students what the sprite draws in the project. 3. Tell them they can click the green flag to see what happens.
10 mins	**To Think** 1. Introduce the Pen feature in Scratch. 2. Explain how to add the Pen extension in Scratch and the usage of the blocks. 3. Show students how to draw a line, change colour, pen up, pen down, etc. 4. Ask students how they might draw a square with Scratch. 1) Ask for volunteers for different approaches. 2) Students demonstrate how to make a square. 5. Guide students to think about which feature they will use and the problem solving procedures.
20 mins	**To Think and To Code** 1. Arouse students to think and work out how to: 1) Draw a Line 2) Draw a Square 3) Use Repeat Block to draw a square 2. Remind students that they should go for testing and debugging after they complete each task.

Lesson 2

Time	Activity
5 mins	**To Play** Let students try another project: https://scratch.mit.edu/projects/737402437/.
30 mins	**To Think and To Code** 1. Let students think about how the snowflake shape is drawn. Ask students to derive the steps in drawing the snowflake (with worksheet). 2. Ask students how they can reuse the same blocks to create a snowflake. Students carry out hands-on practice. 3. Teacher will need to guide students understand about turning 30 degrees after drawing each square in order to complete the snowflake.

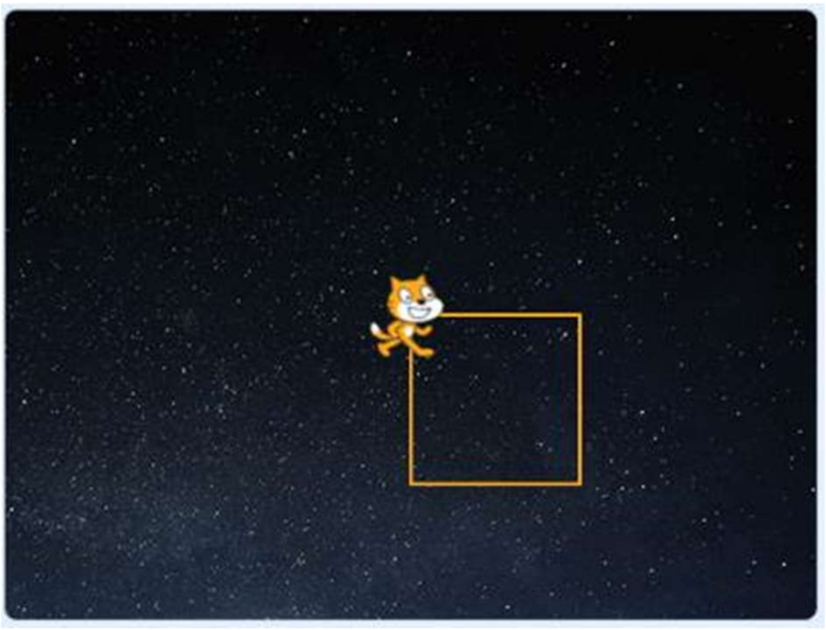
Lesson 3

Time	Activity
5 mins	Recap the concept and skills acquired in last lesson.
20 mins	**To Think and To Code** 1. Ask students to recognize the pattern of "draw a square" and how they can simplify the code. 2. Introduce how to make a custom block, also known as a procedure. Make the "DrawASquare" block. Students should understand the reason for making a custom block. 3. With the help of student guide, students explore making a snowflake shape using a set of rotated squares. The guide will instruct them in learning how to create their own custom block to draw a square.
10 mins	**To Reflect:** **Share the Projects and Provide Constructive Feedback on Program Design** 1. Create a studio and give students the studio URL. Ask students to save and submit their projects to the teacher's Studio. 2. Have one or two students share their project with the whole class. Feedback to be given by peer and teachers. 3. Students should share, think of ways to improve / enhance their program as well as the aesthetic values. **Review of Student Learning** 1. Review on the features of Scratch, and key concepts and practices learnt in the unit. 2. Ask students to complete the review questions, appropriate feedback should be given by teachers.

Drawing Shapes in Scratch

Let's learn to draw shapes in Scratch!

In this unit, you will learn how to draw a line and square using the Pen feature of Scratch.



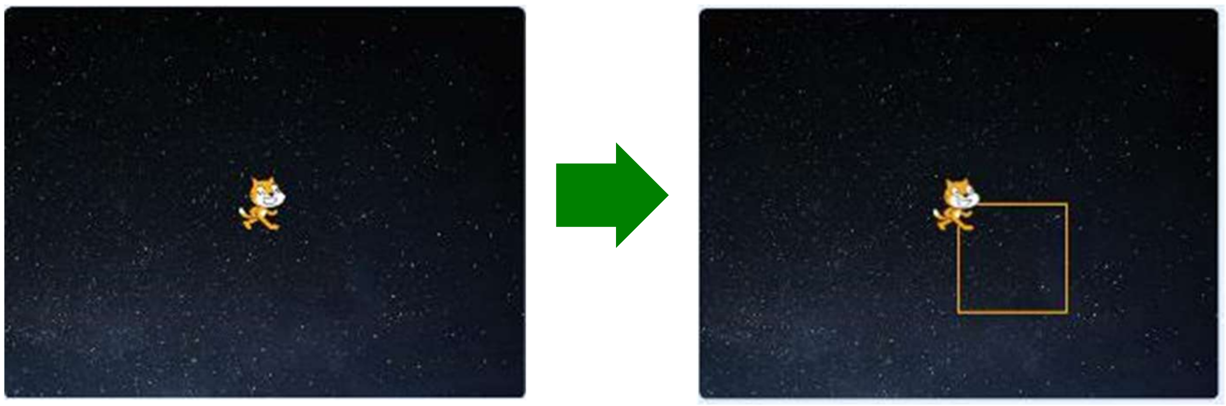
To Play

Open the Scratch Project, Drawing Shapes in Scratch:
https://scratch.mit.edu/projects/737338011/

Click the green flag and see what happen.

Click again to see where the sprite moves.

Did the Scratch cat draw a square?

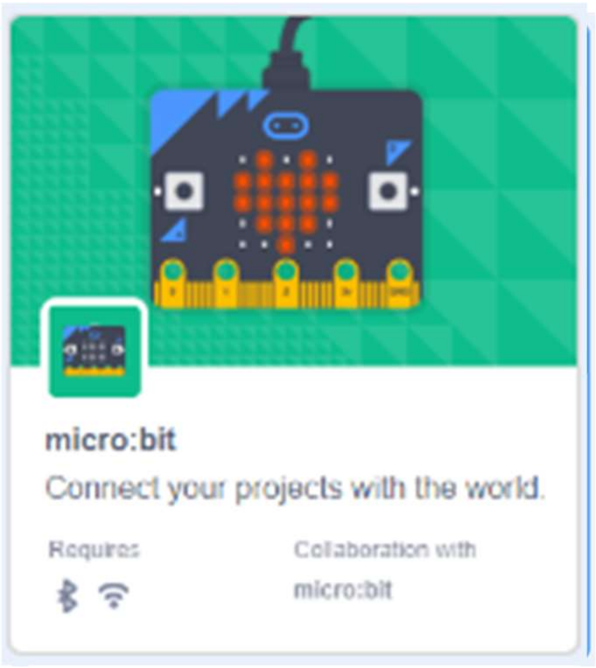


Drawing Shapes in Scratch

To Think

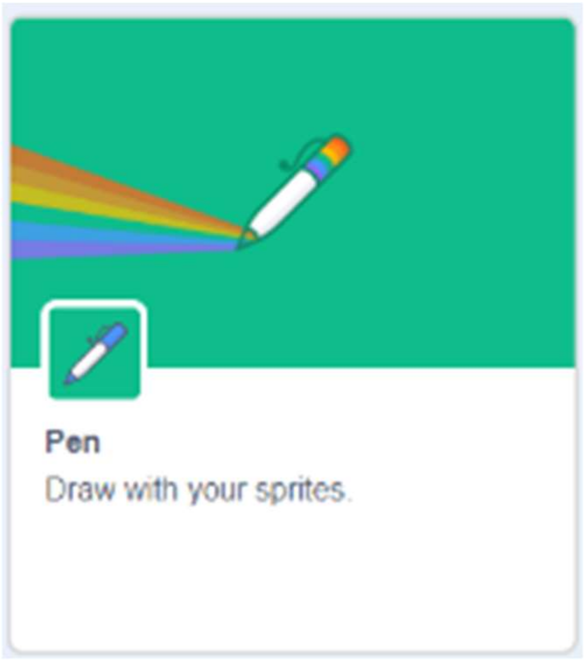
1. Do you know which feature will we use to draw a square in this unit? Tick the box (✓).

❑ Micro:bit



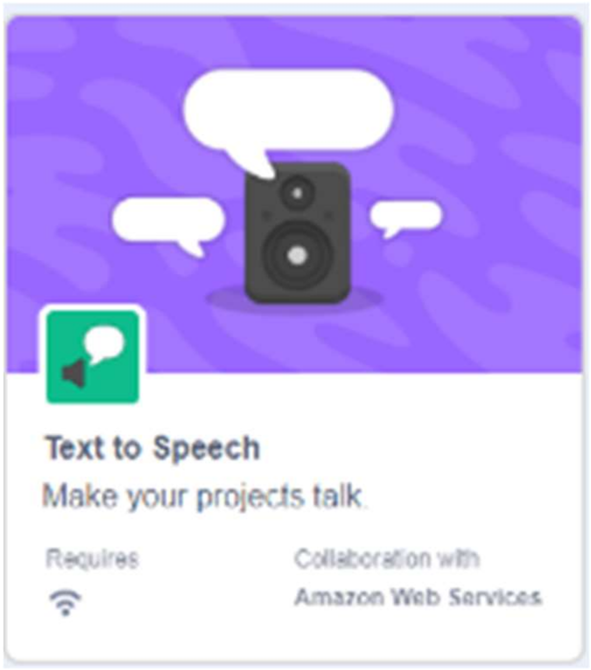
micro:bit
Connect your projects with the world.
Requires
Collaboration with micro:bit

✓ Pen



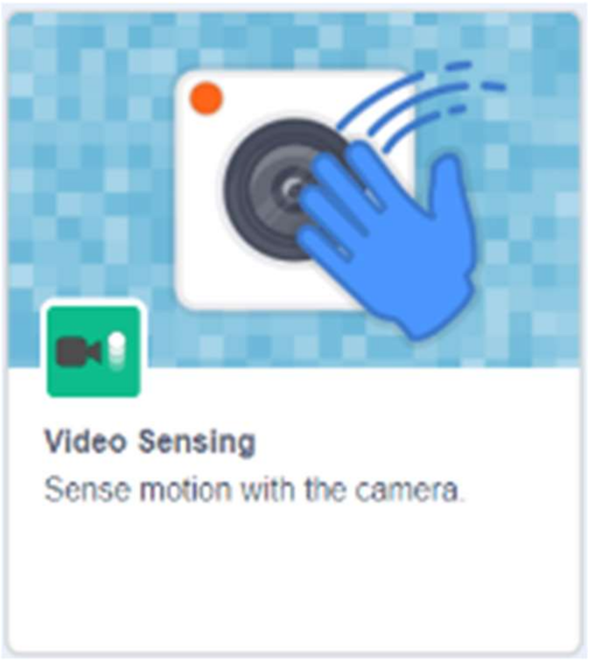
Pen
Draw with your sprites.

❑ Text to Speech



Text to Speech
Make your projects talk.
Requires
Collaboration with Amazon Web Services

❑ Video Sensing



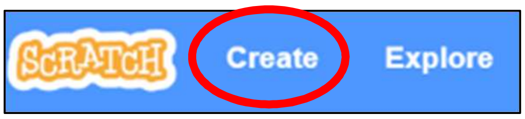
Video Sensing
Sense motion with the camera.

Drawing Shapes in Scratch

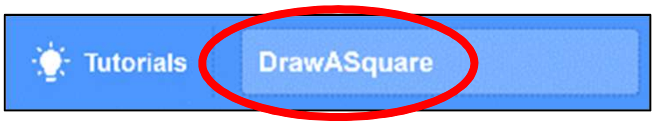
Start Here

1. Sign into your account at https://scratch.mit.edu/ 1

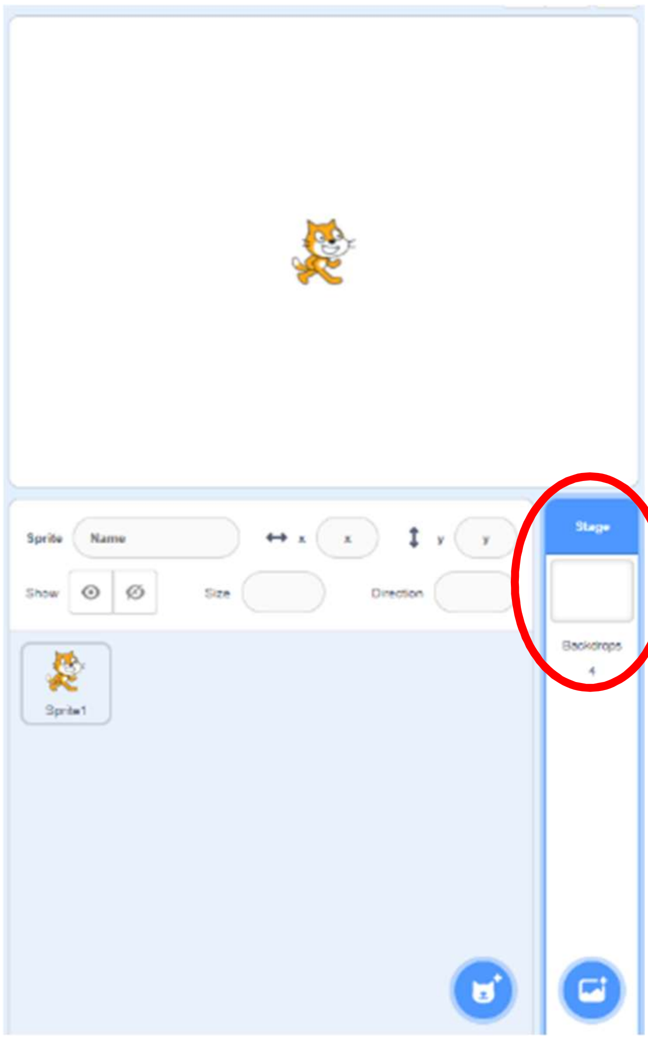
2. Go to "Create" to start a new project. 2



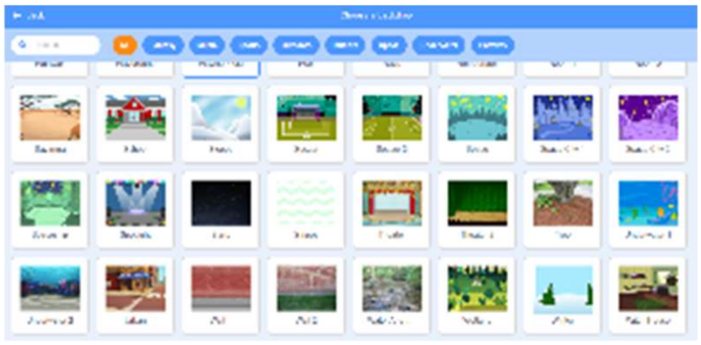
3. Name it "DrawASquare". 3



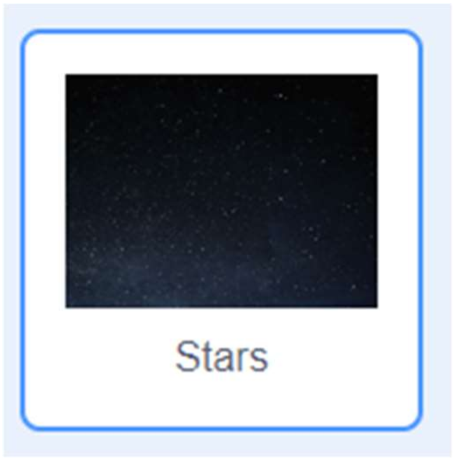
4. You can change the backdrop for your project. 4



5. For example, you may choose "Stars".



5

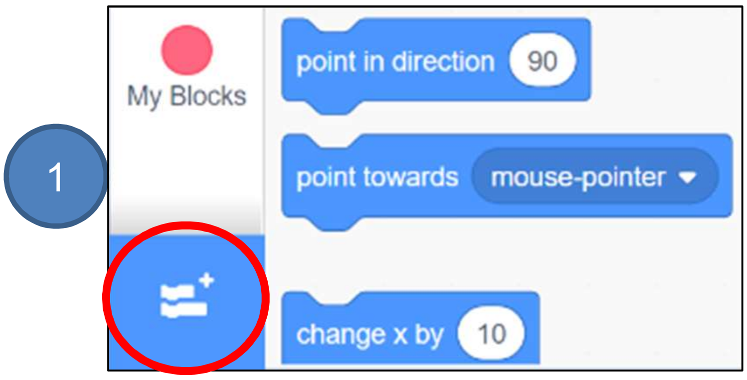


Drawing Shapes in Scratch

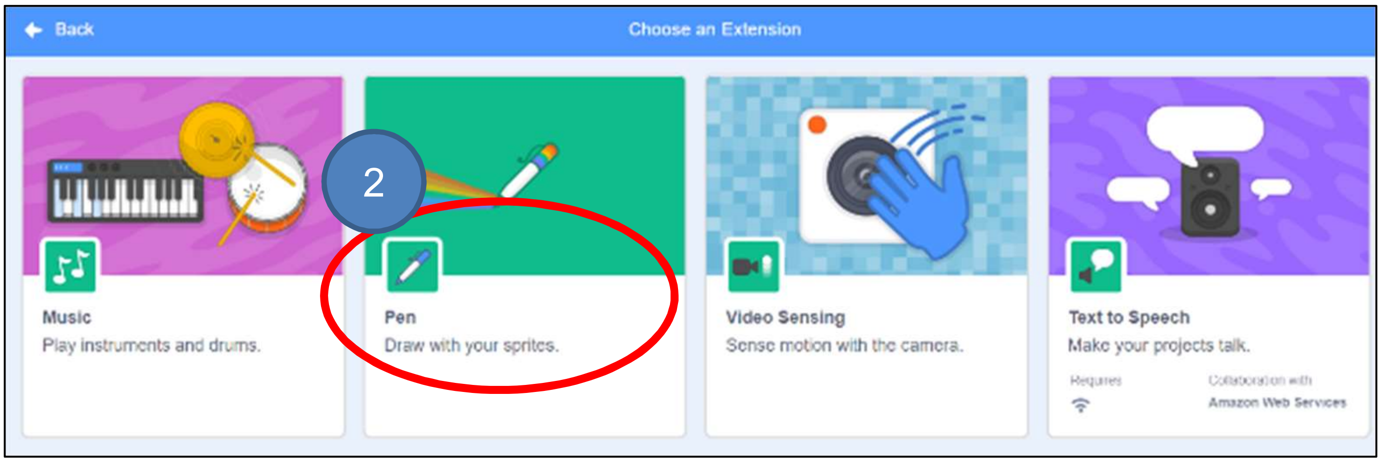
To Code: Add Pen Feature

Before you start drawing, you need to add the Pen component to your Scratch project.

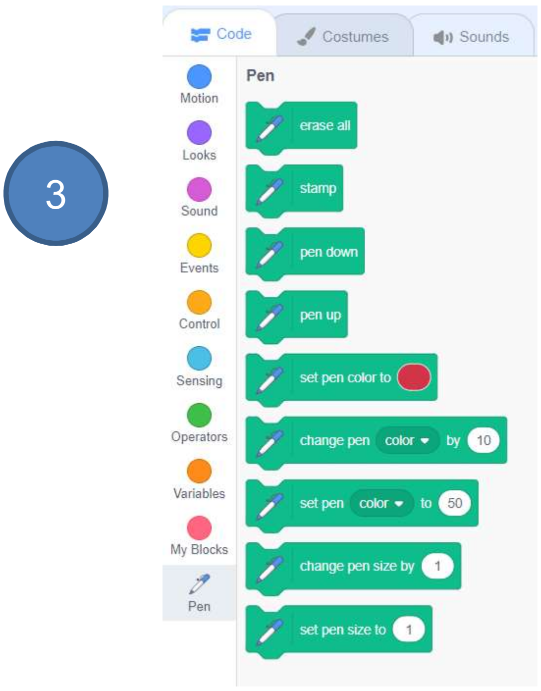
1. Click on the "Add Extension" icon at the bottom left of the page.



2. Select the "Pen" extension.



3. Go to the "Pen" drawer under "Code" tab to see a list of blocks that you can use.

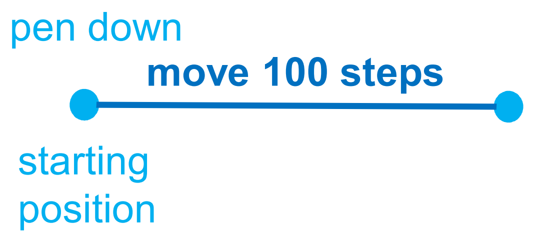


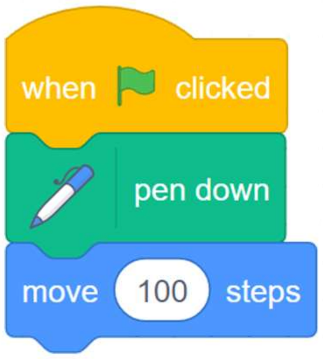
Drawing Shapes in Scratch

See Appendix P.28

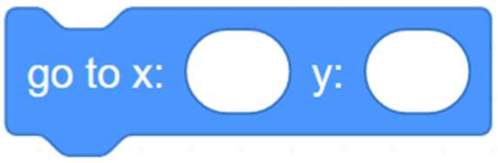
To Code: Draw a Line

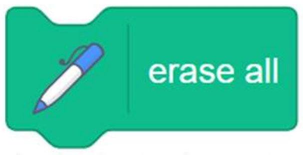
With the Pen feature, by moving the sprite, you can draw a line.



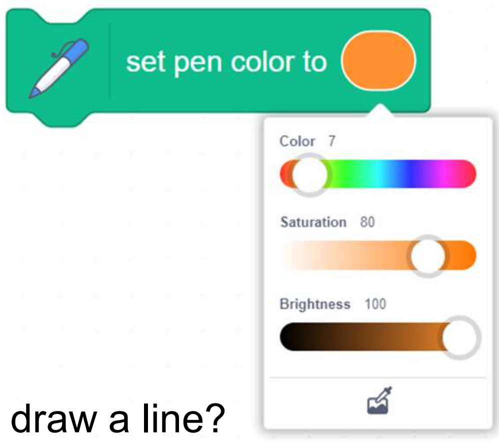


1. Run the code blocks on the right, can you draw a line?
2. Click the green flag again, what is the difference this time?
   How would you solve it?
   Hint: Where is the starting position of the cat?
   What should you do with the marks everytime you draw?





3. Enhancement: Change the pen color to your favourite color.
   You may also change the pen size as well.



Testing and Debugging

Click on the green flag, did you successfully draw a line?

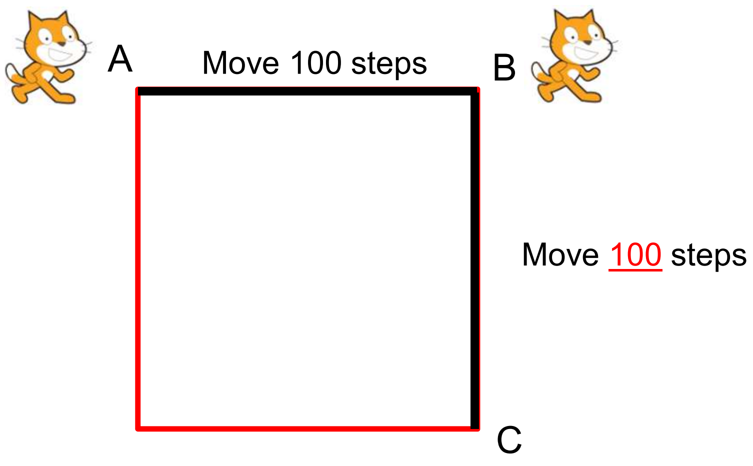
Draw a Line ➡ ➡

Drawing Shapes in Scratch

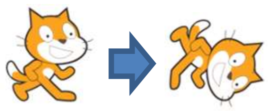
To Think and To Code: Draw a Square

See Appendix P.29

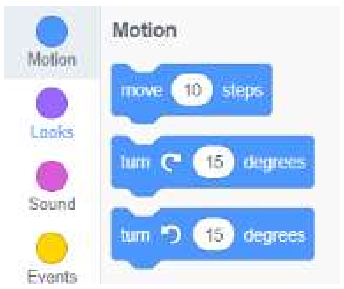
After you drew a line, how can you draw a square?
Before we code, can you continue to draw a square below?
Let's say if the Scratch Cat moves 100 steps from A to B, then how long it moves from B to C?



To move from B to C, the Scratch Cat needs to turn its body down:



Go back to Scratch. In **"Motion"** drawer, drag out a **"turn (right)______degrees"** block.
Hint: Try 30/60/90 degrees for the suitable one with testing.



After you find the right degrees, you may extend the line to a square!

Testing and Debugging

After completing a small step of programming, you can click the green flag and test it to see if your idea works. Then continue programming until you draw a square.

Draw a Line → Draw a Square →

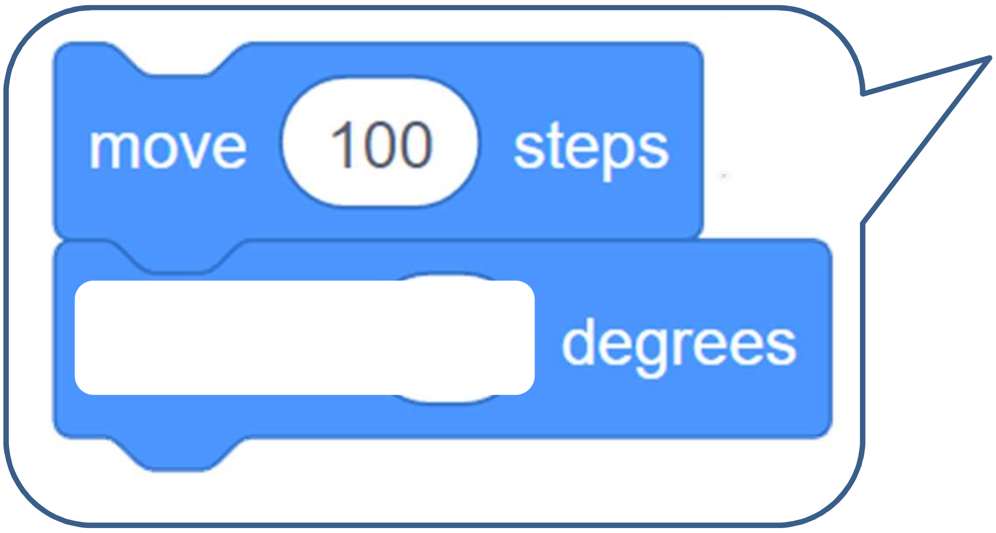
Drawing Shapes in Scratch

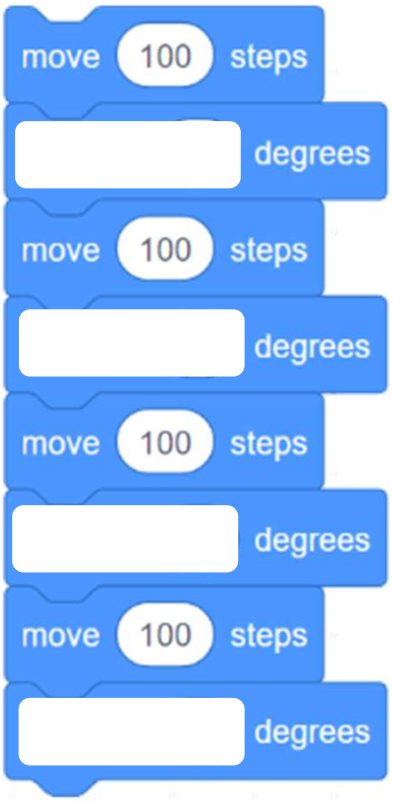
To Think and To Code: Use Repeat Block

See Appendix P.30

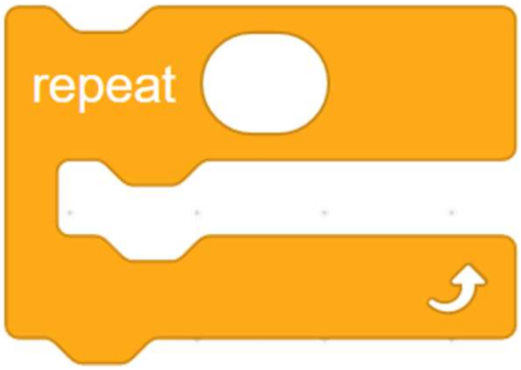
Do you see some blocks are repeating?

How many times do these blocks repeat to draw a square?





Can you find an alternative way to code with repeat block?
Hint: Drag a "repeat" block from the Control drawer.



Testing and Debugging

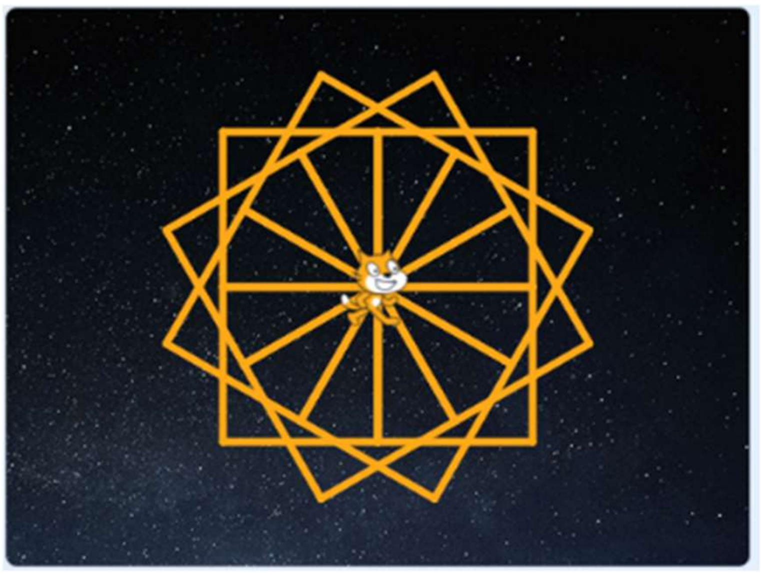
Click on the green flag to test!

Are you able to draw a square using the repeat block?

Draw a Line → Draw a Square → Use Repeat Block

Drawing Shapes in Scratch

In this lesson, you will learn how to make a snowflake with multiple squares.

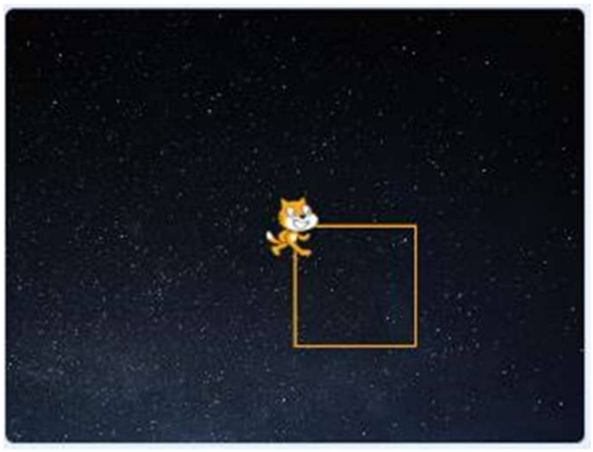


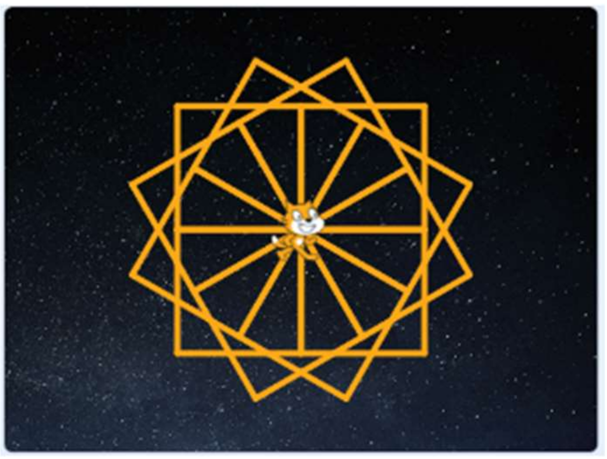
To Play

Open another Scratch Project: https://scratch.mit.edu/projects/737402437

Click the green flag and see what will happen.

Do you know the relationship between a square and a snowflake?

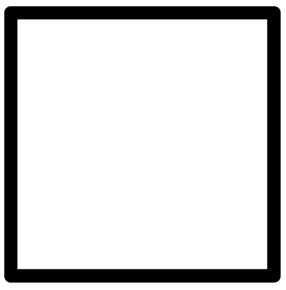




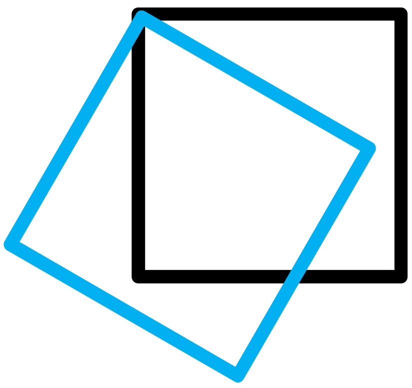
Drawing Shapes in Scratch

To Think: A Snowflake = Multiple Squares

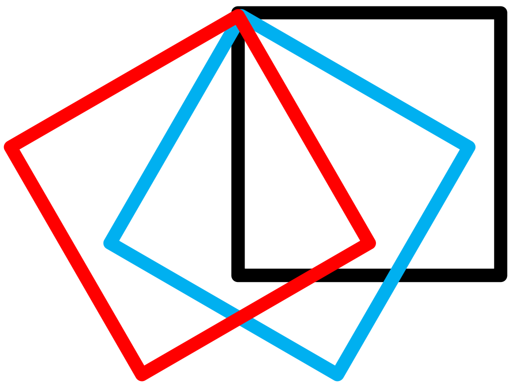
Let's break down the snowflake to see how it can be made.
Fill in the blank to count how many square(s) are in these shapes:



1

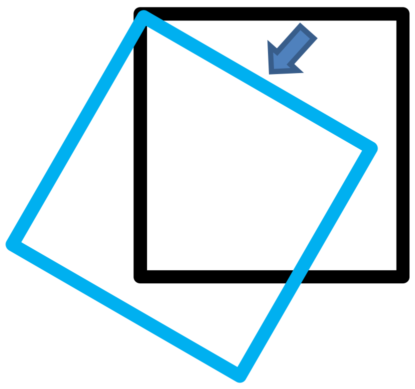


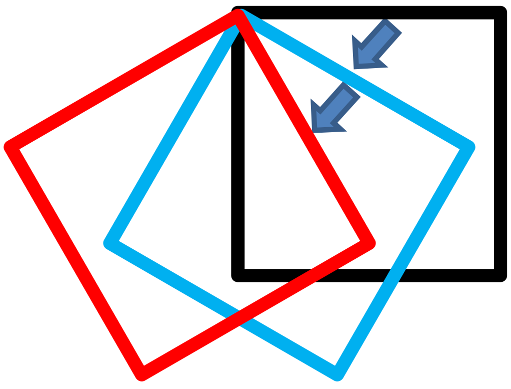
2 ____



3 ____

Draw one square, turn some degrees and then draw another one over and over to make these shapes.



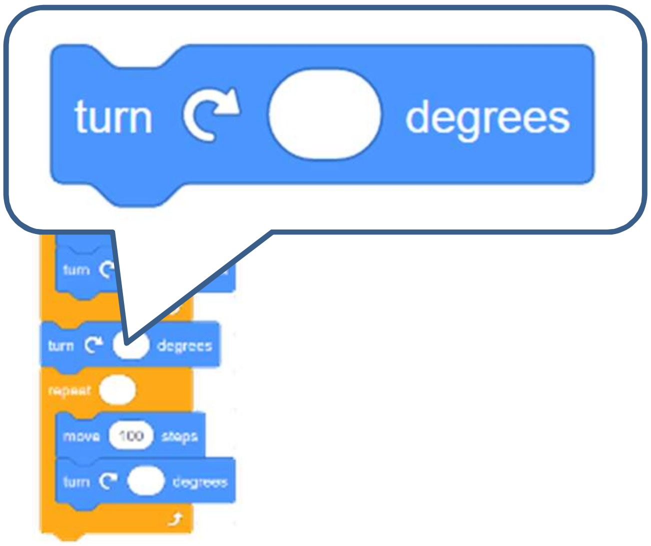


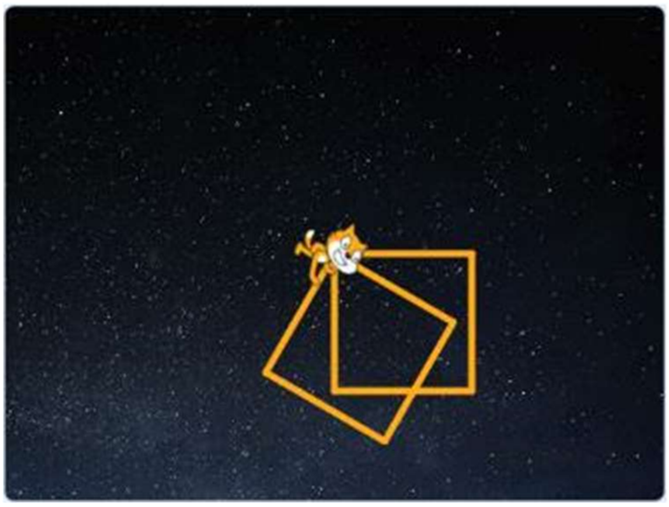
Drawing Shapes in Scratch

To Think and To Code: Draw Multiple Squares

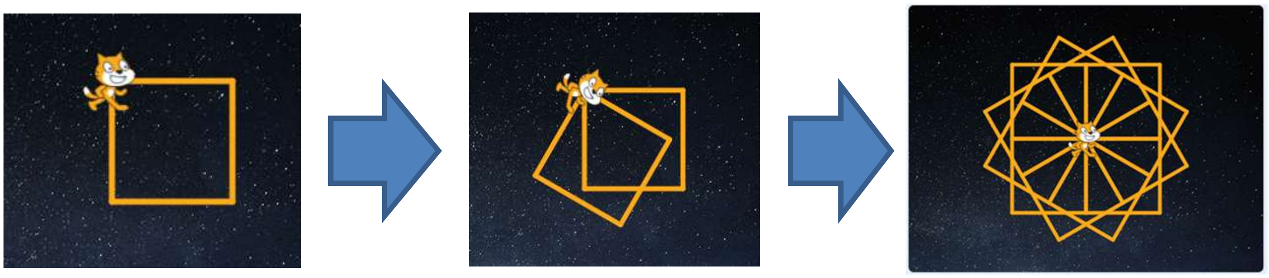
Time to try in Scratch! Save the project as a copy called **DrawASnowflake**.

After drawing the first square, how many degrees do you think it should turn to draw another square? Hint: Try 30 degrees.





Repeat this step (for 12 times) until you can draw a snowflake.



Testing and Debugging

Click the green flag to test! Can you draw multiple squares (a snowflake)?

Drawing Shapes in Scratch

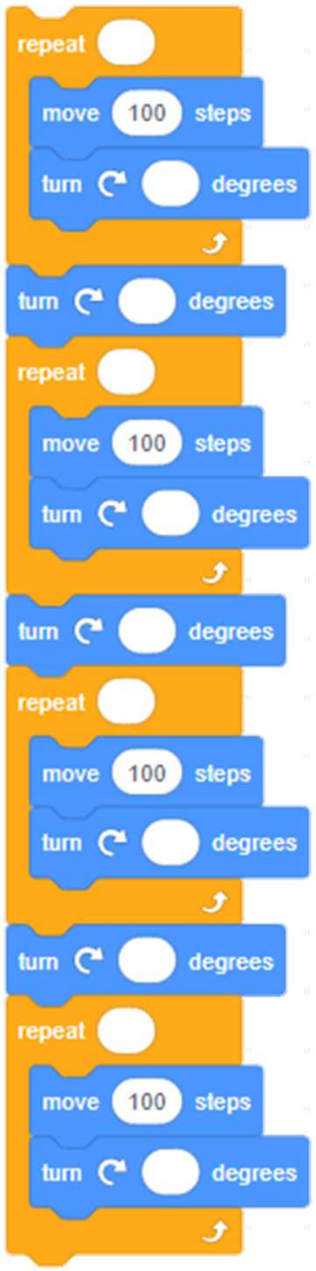
To Think and To Code: Draw Multiple Squares

How long are your code blocks now?
Do you think they are too lengthy?

See Appendix P.31

Based on what you have learnt, how can we make it shorter and clearer?

Hint: Try **“Repeat”** block.



```
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
turn ↻ ( ) degrees
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
turn ↻ ( ) degrees
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
turn ↻ ( ) degrees
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
```

Let’s think:
Can we change the **“Turn___Degrees”** and **“repeat___”** to draw different snowflakes?

Testing and Debugging

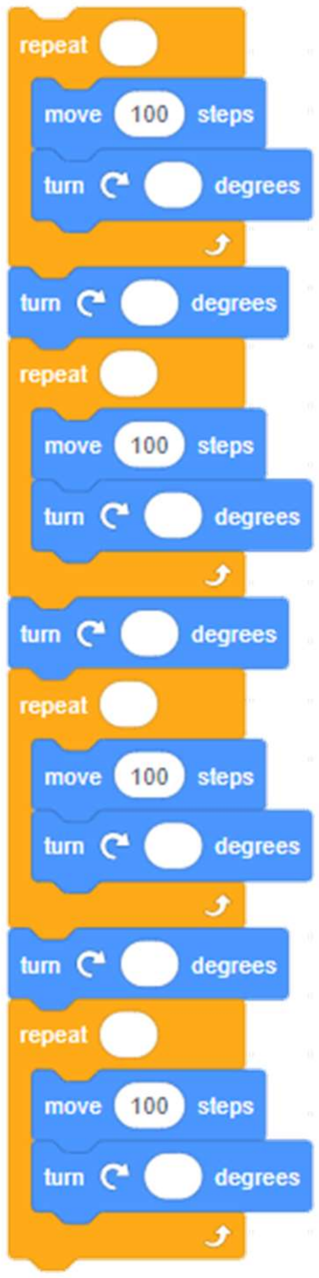
Test again! Can you draw the same shapes with shorter codes?

Drawing Multiple Squares → →

Drawing Shapes in Scratch

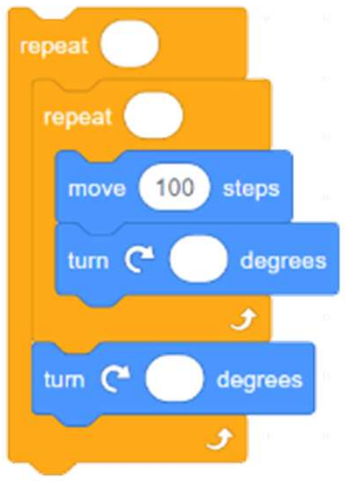
To Think

Check if you use the following way to create two or more squares with shorter blocks:



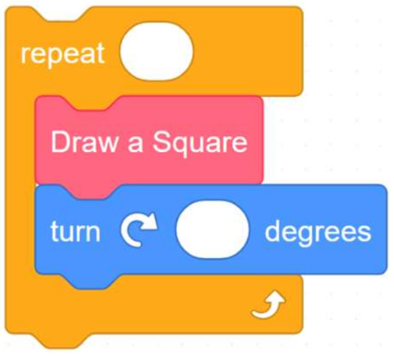
```
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
turn ↻ ( ) degrees
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
turn ↻ ( ) degrees
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
turn ↻ ( ) degrees
repeat ( )
    move (100) steps
    turn ↻ ( ) degrees
```

Is this clear enough?



```
repeat ( )
    repeat ( )
        move (100) steps
        turn ↻ ( ) degrees
    turn ↻ ( ) degrees
```

What if we can just tell computer to "Draw a Square"?



```
repeat ( )
    Draw a Square
    turn ↻ ( ) degrees
```

Drawing Shapes in Scratch

To Think

Unplugged Activity: Drinking Water

Drinking water is simple. Can you write down the steps of “drinking water”?

A

B

Drinking Water
Take out the water bottle, unscrew the cap, point it at your mouth, pour out the water...

Usually, when you want to drink water in the classroom, how would you ask the teacher? Try these two different ways to do so:

A. Teacher, can I “drink water”?

B. Teacher, can I (all the steps of “drinking water”)?

Which way is clear and better?

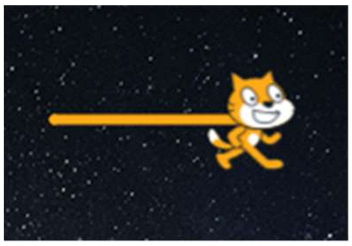
In the same way, how should we tell computer to “Draw a Square”? Recall the code blocks you used to draw the square in Scratch. What are the detailed steps?

Asking the computer to “Draw a Square" is like "drinking water". First teach it all the detailed steps of drawing a square (draw a line, turn 90 degrees, repeat 4 times), and then use “My Blocks” in Scratch to build a module that abstracts these actions into “Draw a Square".

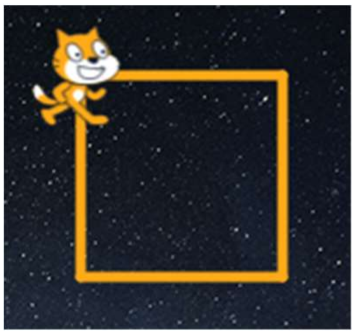
Drawing Shapes in Scratch

To Think

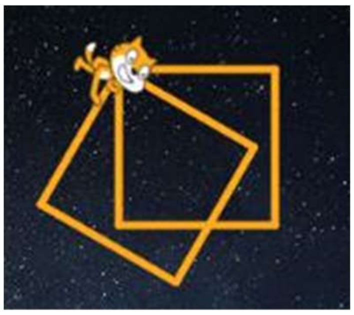
Can you write down the steps that we draw a line/square/multiple squares?



Draw a **Line**:
Move 100 steps



Draw a **Square**:
Repeat the steps of "Draw a Line" > Turn 90 Degrees > Repeat 4 Times



Draw **Multiple Squares**:
Repeat the steps of "Draw a Square" > Turn 30 Degrees > Repeat 2 (or more) Times

Suggested answer only.
Students may have different answers based on their snowflakes in different styles (how many squares they drew at this point).

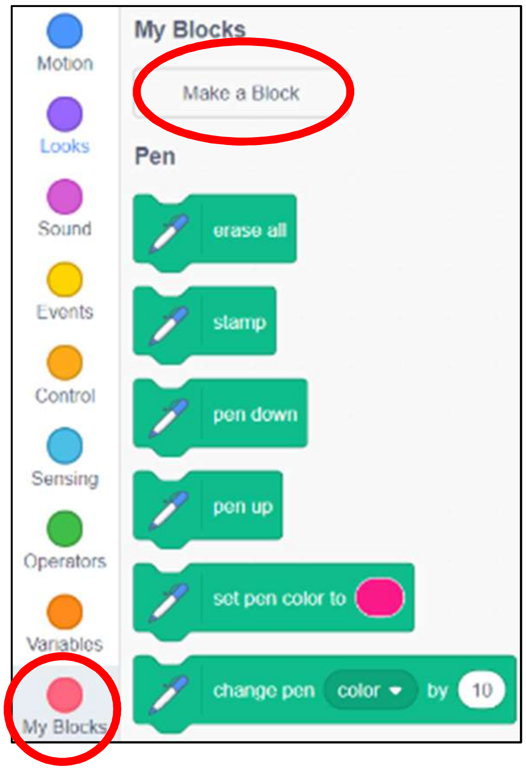
Drawing Shapes in Scratch

To Think and To Code: Create "DrawASquare" Block

Let's **make** our own custom block!

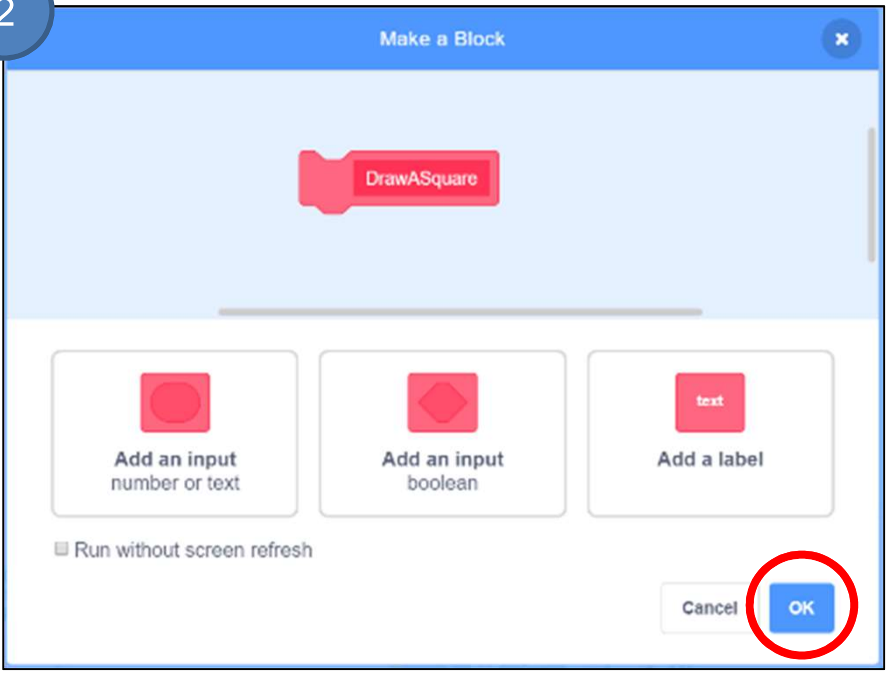
1. Click on the **"Make a Block"** button in the **"My Blocks"** drawer.

1



2. Type **"DrawASquare"** as the name of the block and then **"OK"**.

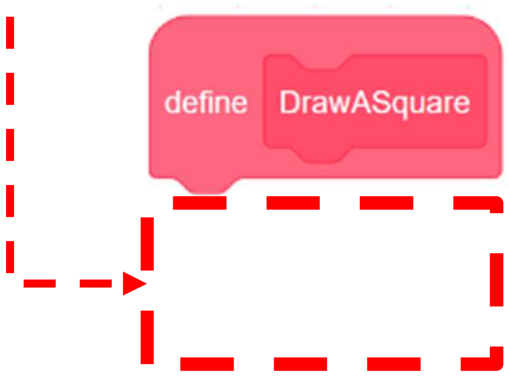
2



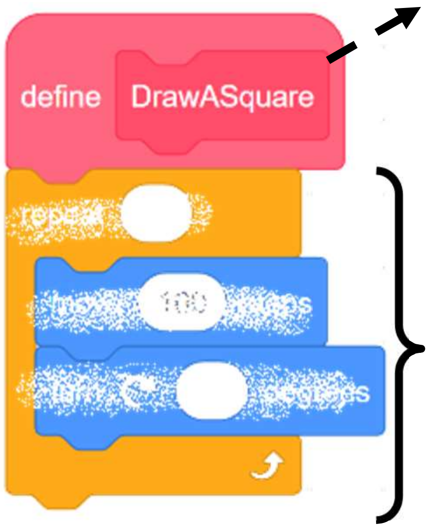
Drawing Shapes in Scratch

To Think and To Code: Create "DrawASquare" Block

After creating the own custom block, **DrawASquare**, you will now see a define DrawASqaure block.
Think about which blocks that draw a square and snap them to it. That means to **"define DrawASquare"**.



Knowledge builds up: Abstraction and Modularization



Abstraction is the process of identifying key information that is relevant in a given context and forgetting those details in that context.
For example, the name the custom block (procedure) "DrawASquare" captures its abstract key meaning of what to do later in the module.

Modularization is the process of organizing code for the task it executes and always try to keep the code reusable. It can be written once and be used in multiple places in the program. For example, we can use the "DrawASquare" module repeatedly to draw squares.

Drawing Shapes in Scratch

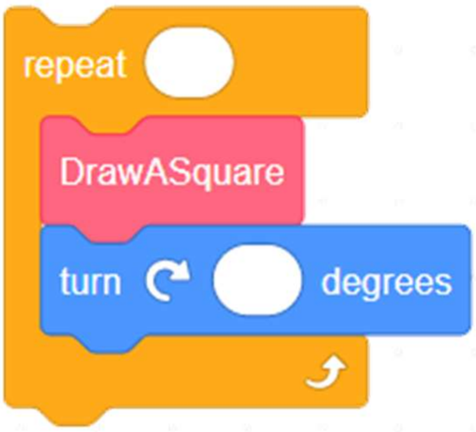
To Think and To Code: Create "DrawASquare" Block

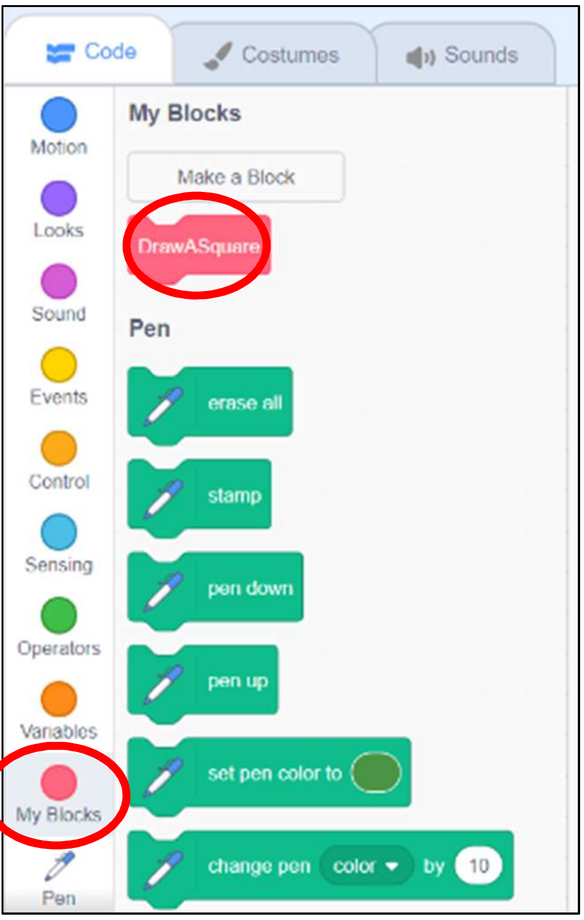
See Appendix P.32

Creating a custom block does not mean that it will run by itself; therefore we need to call it.

1. In the **"My Blocks"** drawer, find and drag the **"DrawASquare"** block out.

2. Then, with our custom block, **draw multiple squares.**





Testing and Debugging

When you finish, you can test it at once! Can you draw a snowflake with our custom block?

Drawing Multiple Squares → Create "DrawASquare" Block →

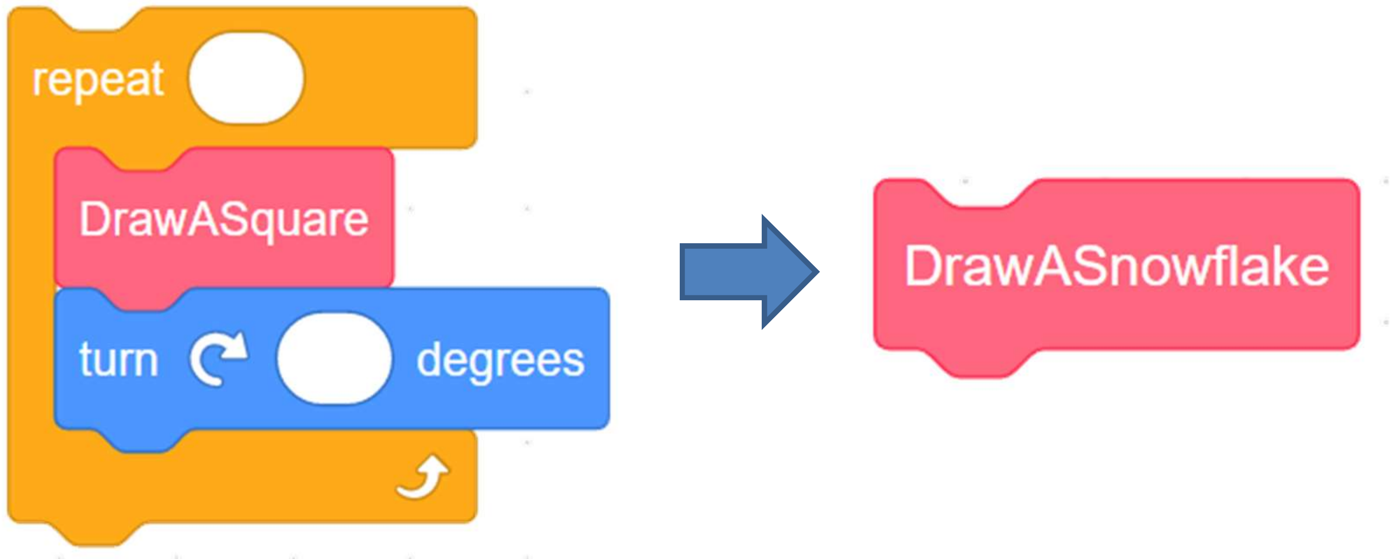
Drawing Shapes in Scratch

To Think and To Code: Create “DrawASnowflake” Block

See Appendix P.33

Finally, what if we want computer to “Draw a Snowflake”?

Review your code blocks and create another block named “**DrawASnowflake**” to do so.



Testing and Debugging

Click the green flag and test to see if the snowflake is still the same.

Drawing Multiple Squares → Create “DrawASquare” Block → Create “DrawASnowflake” Block

Drawing Shapes in Scratch

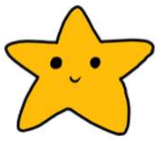
To Reflect: Two Stars and a Wish Worksheet

Name of Project: ____________ Name of Creator: ____________

Please write down two things that you like about this project.

“Two Stars and a Wish” is a reflection strategy designed for student feedback as peer- and self-assessment.
Teachers can guide students to give constructive feedback to their peers regarding their Scratch project - two positive (stars) and one hopeful (wish) reflection. Comments can be made on Scratch project’s idea, features and aesthetic aspects etc.

1 ____


2 ____

What is one thing you would like to add or change to make this project better?

____

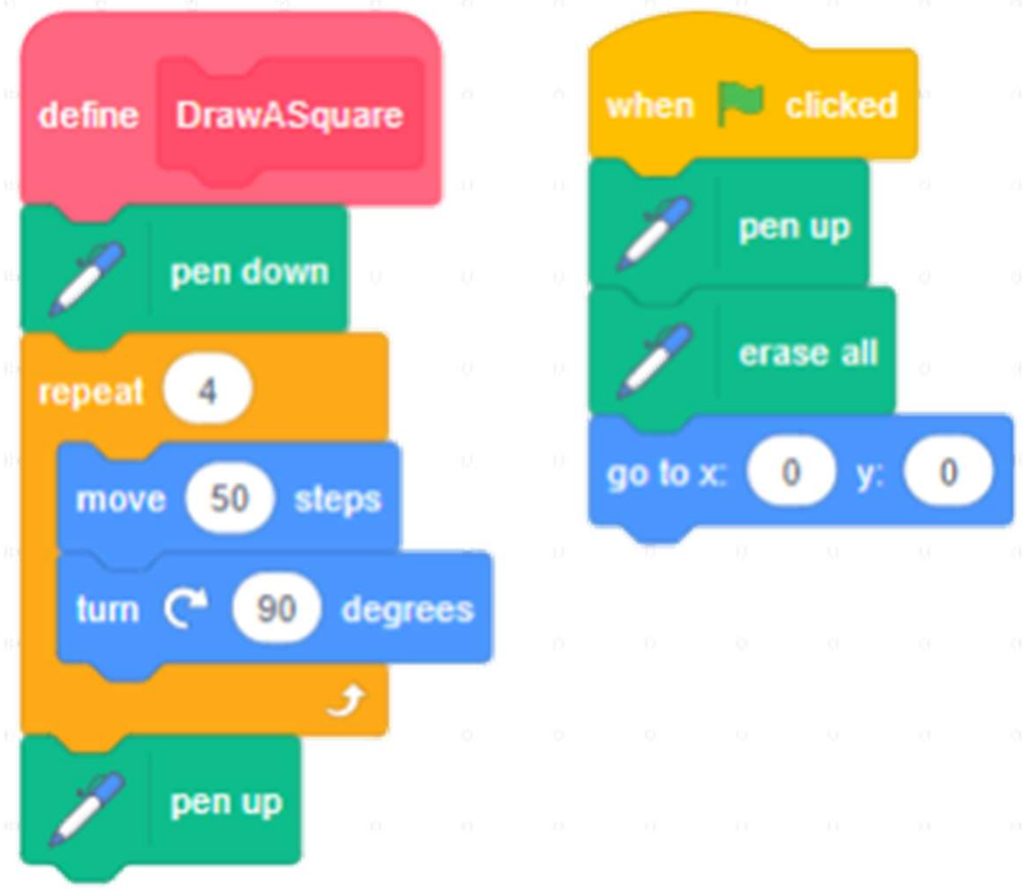
____

____

Drawing Shapes in Scratch

Review Questions

1. A student makes a project with the blocks above. What happens when the user clicks the green flag?



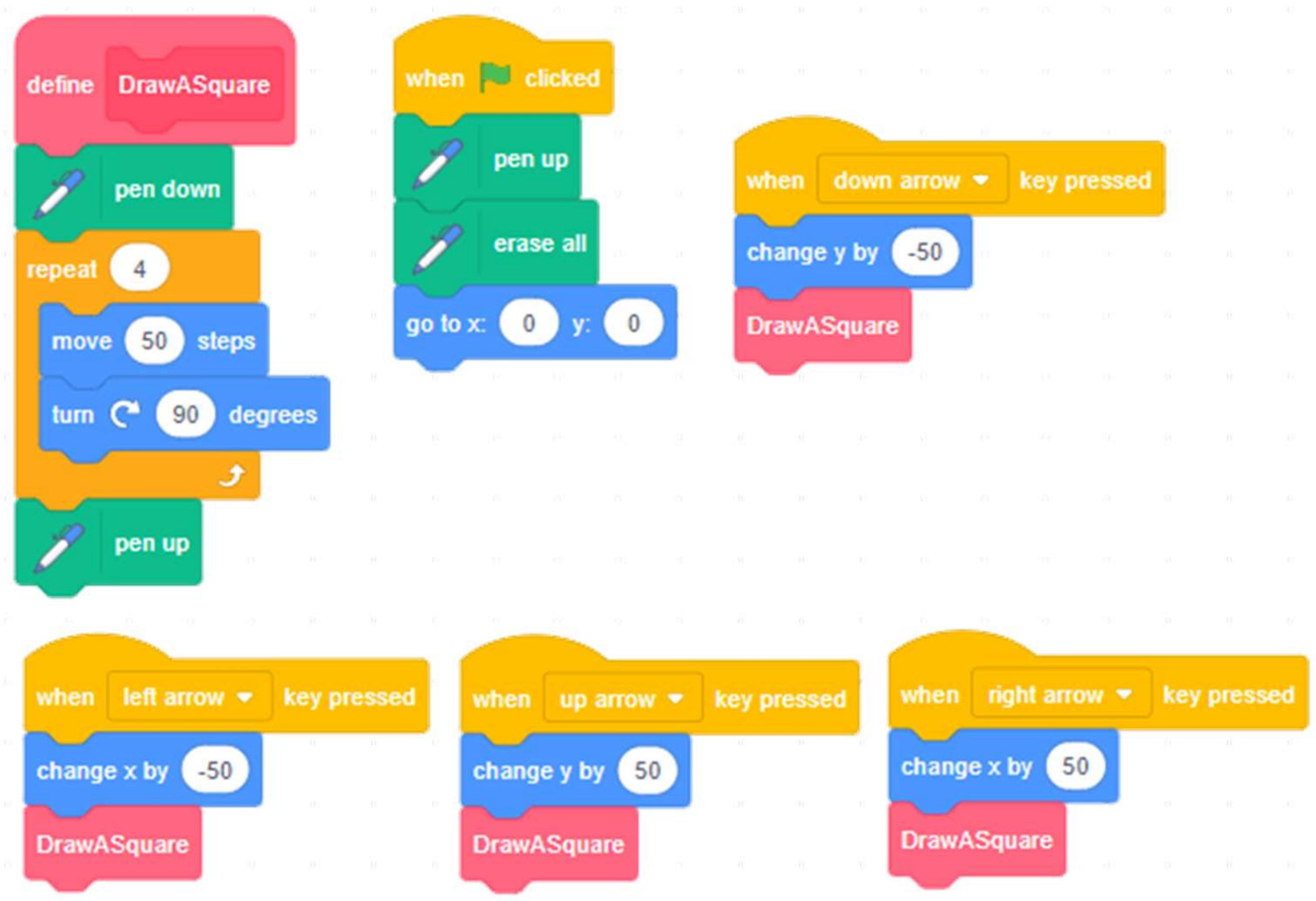
A. The sprite draws a snowflake.
B. The sprite draws four squares.
C. The sprite draws one square.
D. The stage is cleared and the sprite moves to the centre of the stage.

(Answer: D)

Drawing Shapes in Scratch

Review Questions

2. What happens if the user presses the right arrow 3 times?



```
define DrawASquare
pen down
repeat 4
    move 50 steps
    turn ↻ 90 degrees
pen up

when ⚑ clicked
pen up
erase all
go to x: 0 y: 0

when down arrow key pressed
change y by -50
DrawASquare

when left arrow key pressed
change x by -50
DrawASquare

when up arrow key pressed
change y by 50
DrawASquare

when right arrow key pressed
change x by 50
DrawASquare
```

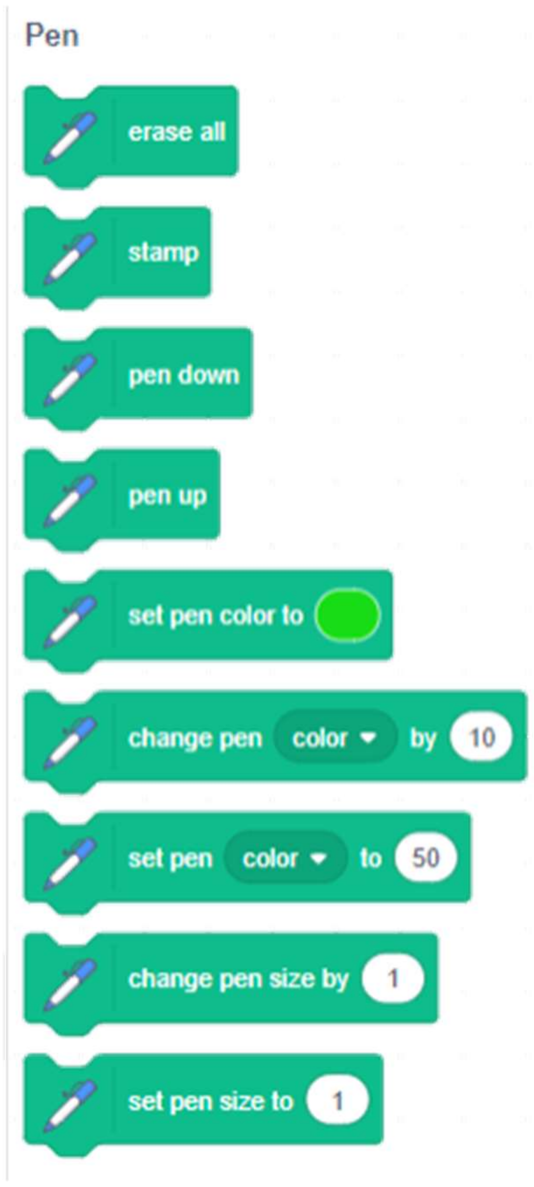
A. The sprite moves to the right 30 steps.
B. The sprite moves to the centre of the stage.
C. The sprite draws 3 squares that are beside each other on the stage.
D. The sprite draws 3 squares that are above each other on the stage.

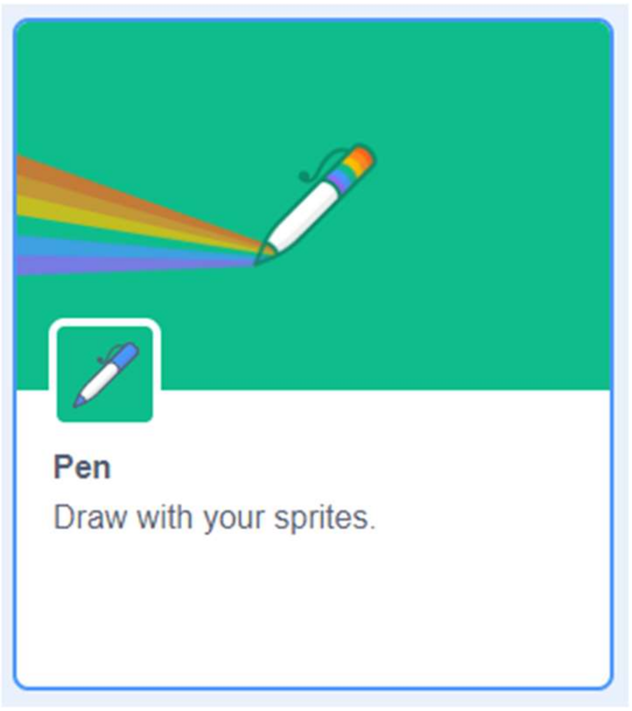
(Answer: C)

Drawing Shapes in Scratch

Revision on Key Features

Pen: Turn the sprite (Scratch Cat) into a pen and draw on the stage freely.

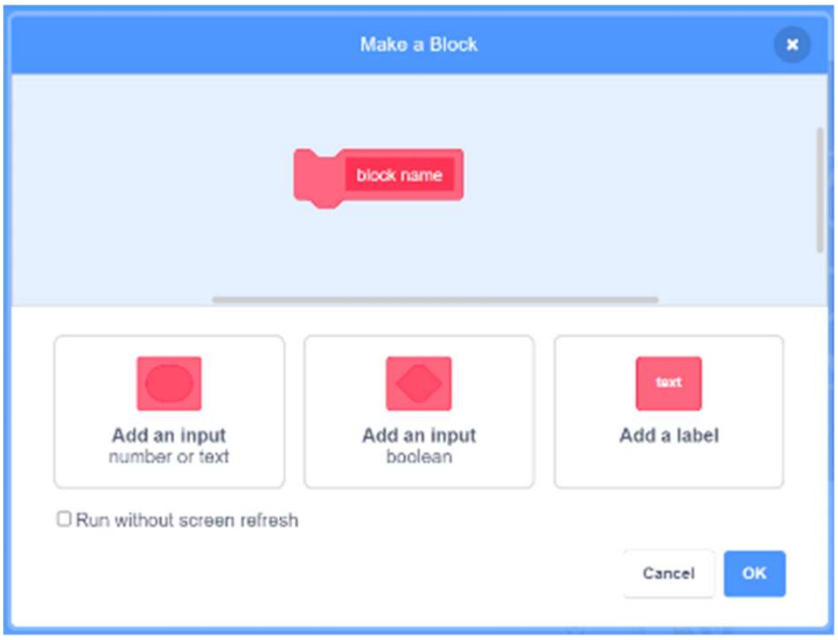


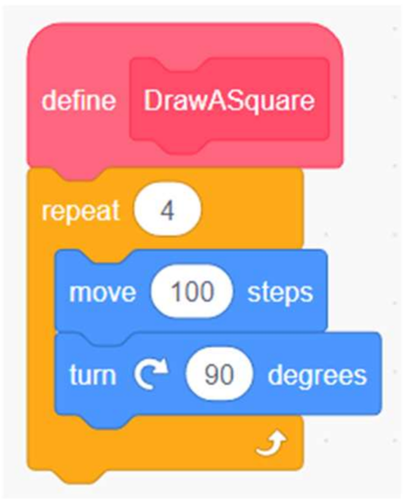


Drawing Shapes in Scratch

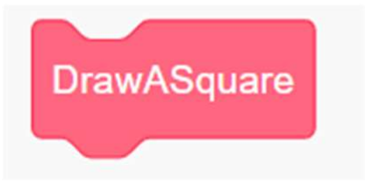
Revision on Key Features

My Blocks: create code blocks to modularize and abstract sequences of commands.





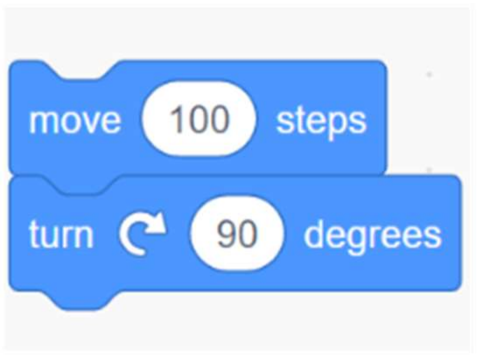
```
define DrawASquare
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
```

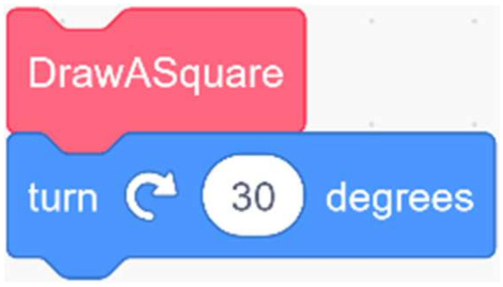


Drawing Shapes in Scratch

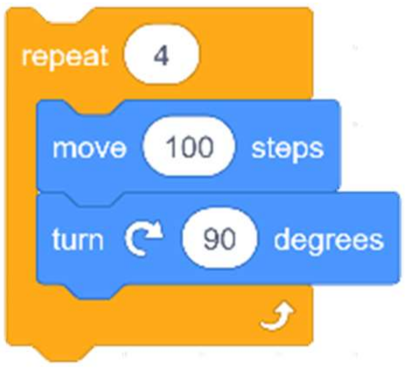
Revision on Key Concepts & Practices

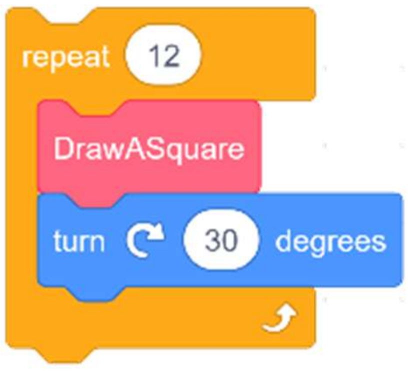
Sequences: A series of steps that are executed by the computer one after the other in an order.





Iteration: Iteration is repeating a process in order to produce a sequence of outcomes. “Repeat _____” blocks can trigger iteration in Scratch.





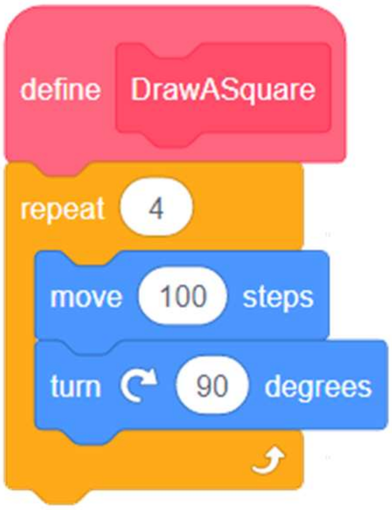
Drawing Shapes in Scratch

Revision on Key Concepts & Practices

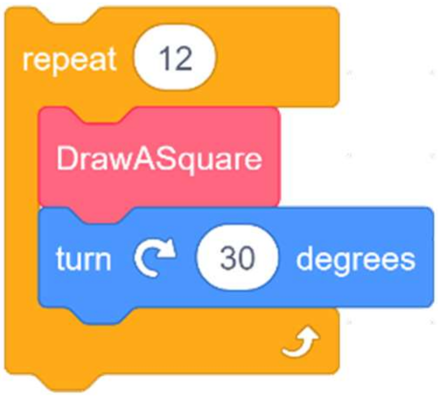
Abstraction and Modularization:

In computer programming, Abstraction is the process of identifying key information that is relevant in a given context and forgetting those details in that context. For example, the name the custom block (procedure) “DrawASquare” capture its abstract key meaning of what to do later in the module.

In computer programming, Modularization is the process of organizing code for the task it executes and always try to keep the code reusable. For example, we can use the “DrawASquare” module repeatedly to draw squares.



```
define DrawASquare
repeat 4
    move 100 steps
    turn ↻ 90 degrees
```

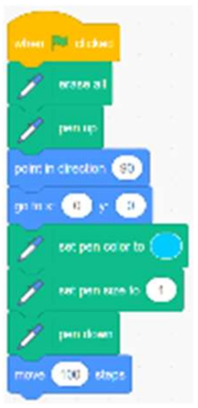


```
repeat 12
    DrawASquare
    turn ↻ 30 degrees
```

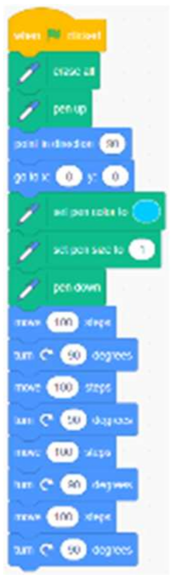
Drawing Shapes in Scratch

Revision on Key Concepts & Practices

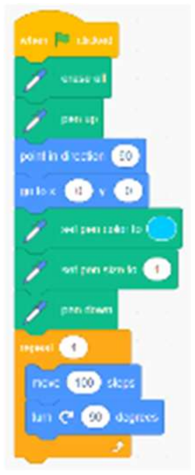
Being incremental and iterative: to work out a sub-task as an iteration, try it out, then work out another sub-task in one more iteration until the whole programming task is completed.



```
when 🏴 clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to ( )
set pen size to (1)
pen down
move (100) steps
```

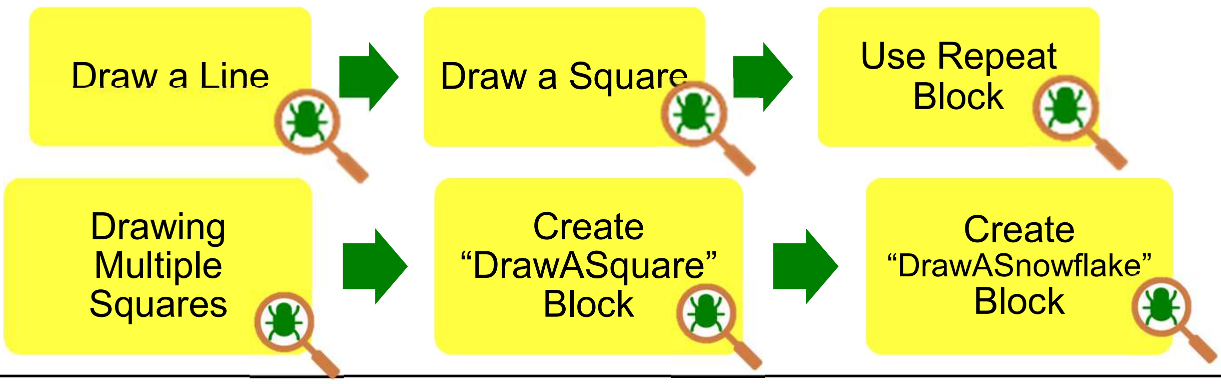


```
when 🏴 clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to ( )
set pen size to (1)
pen down
move (100) steps
turn ↻ (90) degrees
move (100) steps
turn ↻ (90) degrees
move (100) steps
turn ↻ (90) degrees
move (100) steps
turn ↻ (90) degrees
```



```
when 🏴 clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to ( )
set pen size to (1)
pen down
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
```

Testing and Debugging: Testing a computer program is the process of checking if it can produce outcomes as designed. Debugging a computer program is the process of finding out ways to revise the program so that the bugs can be removed.



Appendix

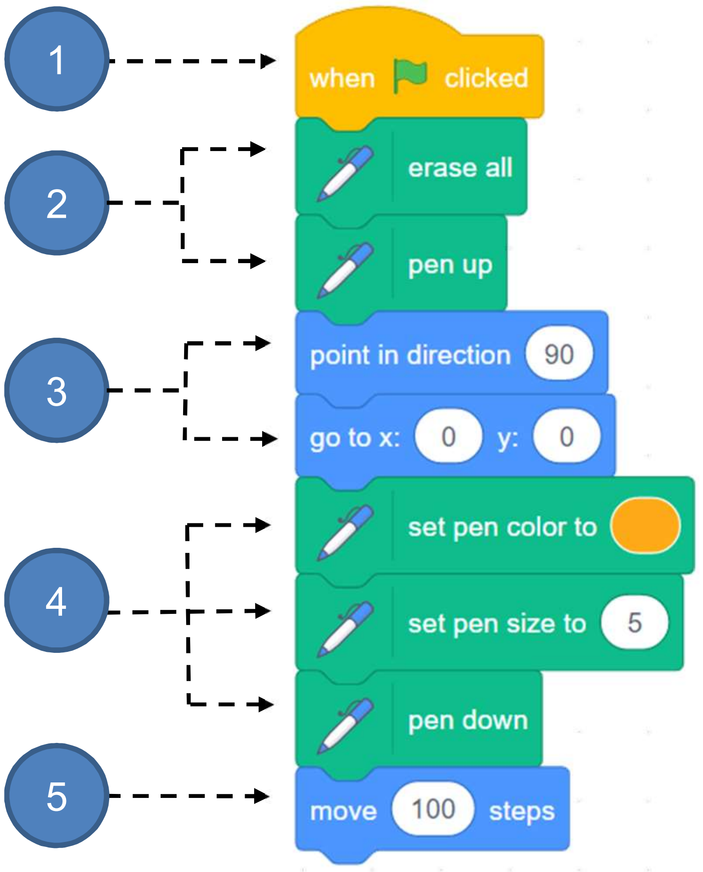
Operation Manual

Drawing Shapes in Scratch

To Code: Draw a Line

See Teacher Guide P.5

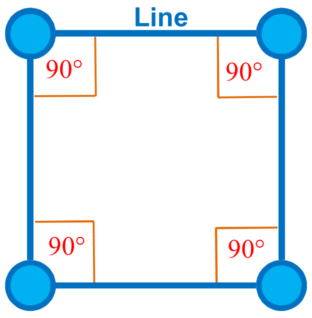
1. Drag the **"when green flag clicked"** block out from the **Events** drawer.
2. To clear the screen, drag out the **"erase all"** and **" pen up "** blocks from the **Pen** drawer and snap them to the **" when green flag clicked "** block. The **" pen up "** block prevents the sprite from drawing while its moving.
3. To get the sprite to its starting position, drag out the **"point in direction 90 "** and **" go to x:0 y:0 "** blocks from the **"Motion"** drawer and snap to the above blocks.
4. Drag out the **"set pen color to", " set pen size to"** and **"pen down"** blocks from the **Pen** drawer. Snap them to the above blocks to get the sprite ready to draw.
5. Add a **"move 100 steps"** block from the **"Motion"** drawer to draw the line.



Drawing Shapes in Scratch

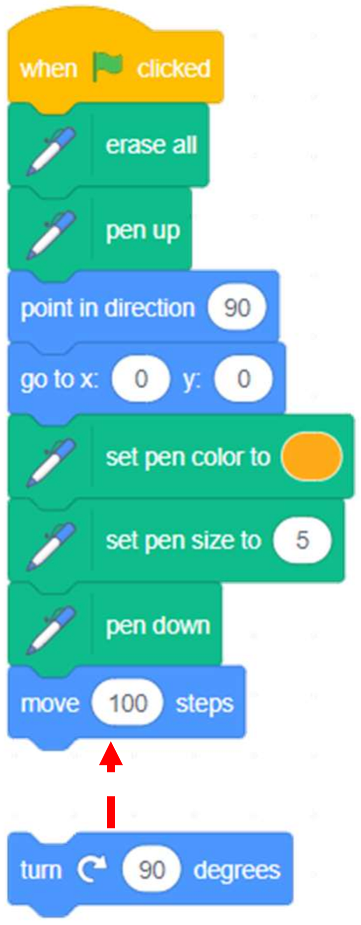
To Think and To Code: Draw a Square

A square is created by joining four lines at right angles (90 degrees).

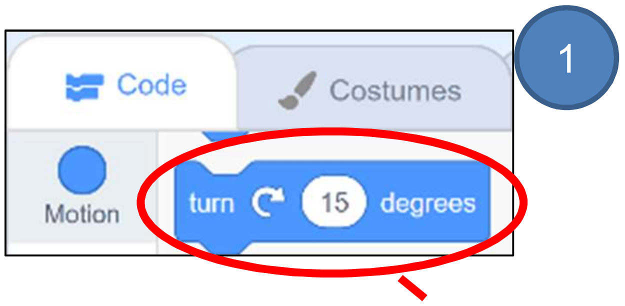


See Teacher Guide P.6

To draw a square, you need to turn 90 degrees before you start to draw the next line.

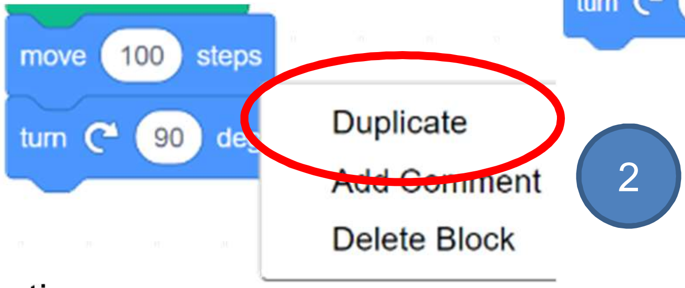


1. Snap a **turn right 90 degrees** block from the **Motion** drawer to the **move 100 steps** block.



Change to 90 degrees

2. Right-click on the **move 100 steps** block to duplicate the **move 100 steps** and **turn right 90 degrees** blocks. Snap the duplicate blocks below the blocks created in step 1.



3. Repeat **step 2** two more times.

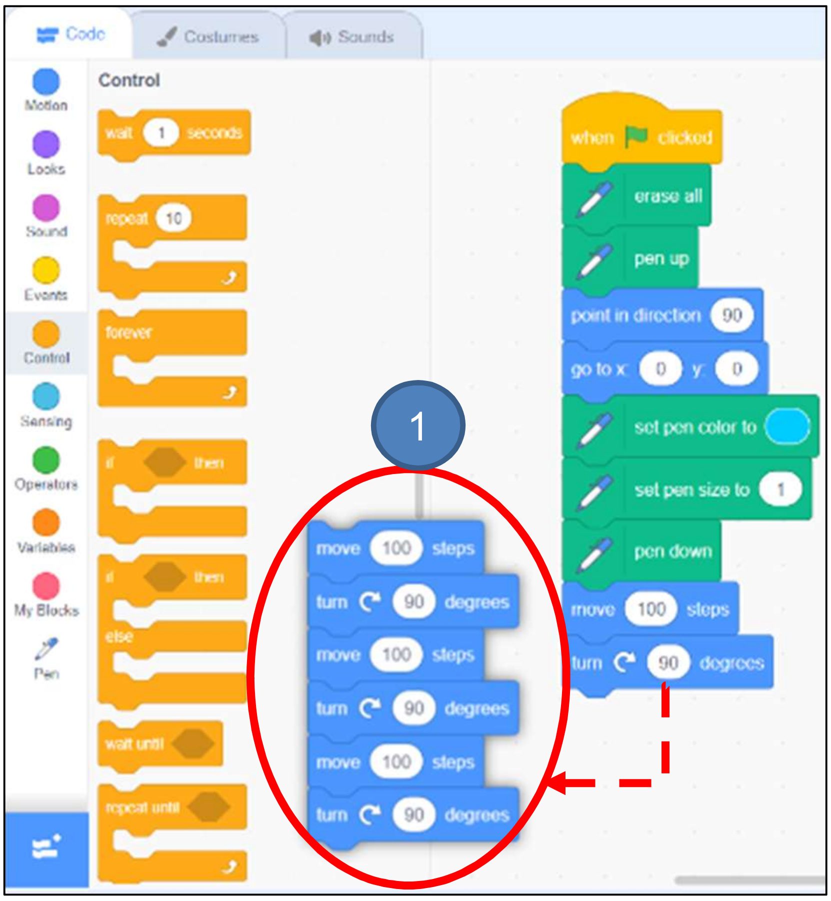
3

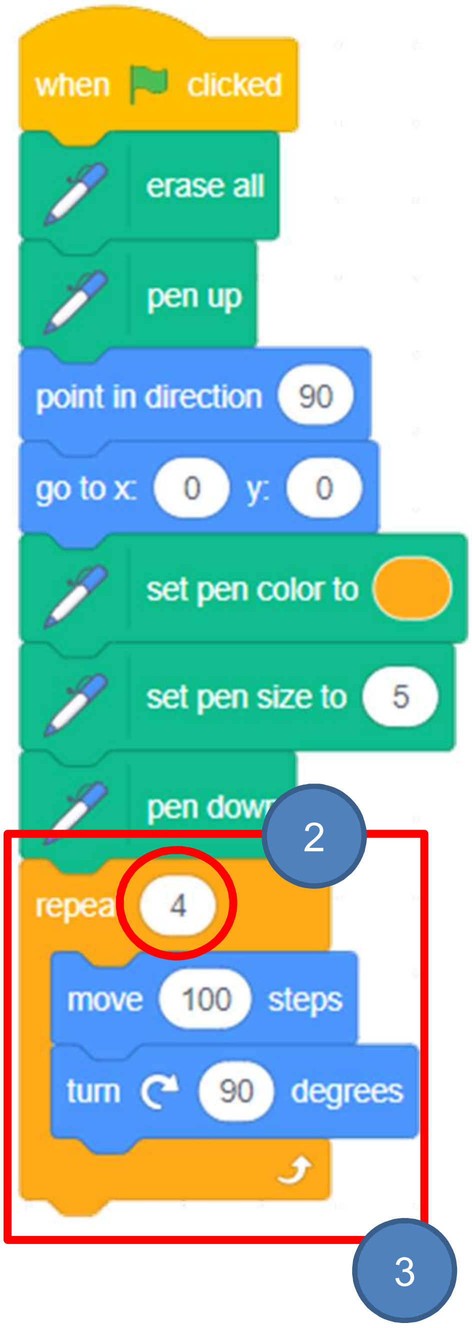
Drawing Shapes in Scratch

To Think and To Code: Use Repeat Block

See Teacher Guide P.7

1. Remove the three sets of duplicate blocks: **move 100 steps** and **turn right 90 degrees**.





2. Drag a **repeat** block from the **Control** drawer and change the number to “4”.

3. Snap the remaining **move 100 steps** and **turn right 90 degrees** blocks into the **repeat** block.

Drawing Shapes in Scratch

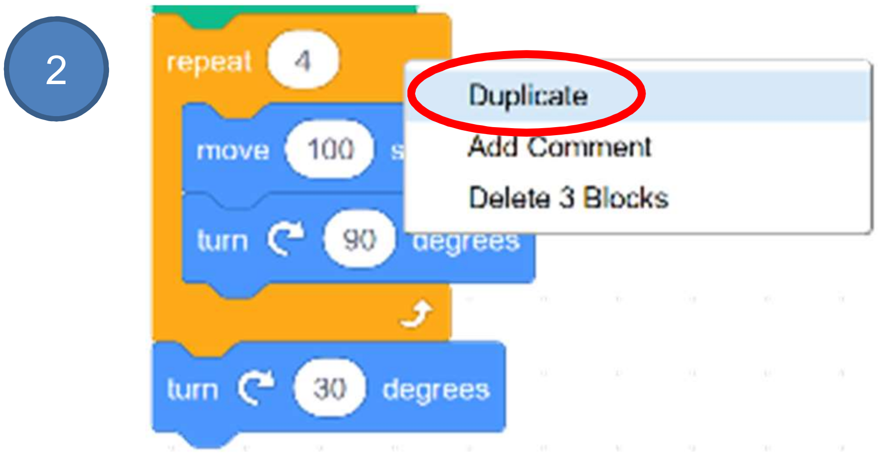
To Think and To Code: Draw Multiple Squares

To draw a snowflake with multiple squares, you need to:

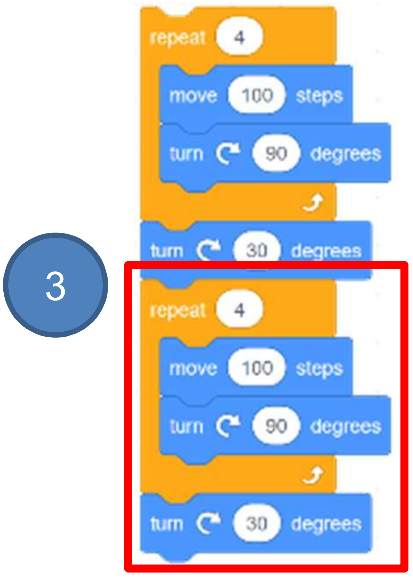
See Teacher Guide P.11

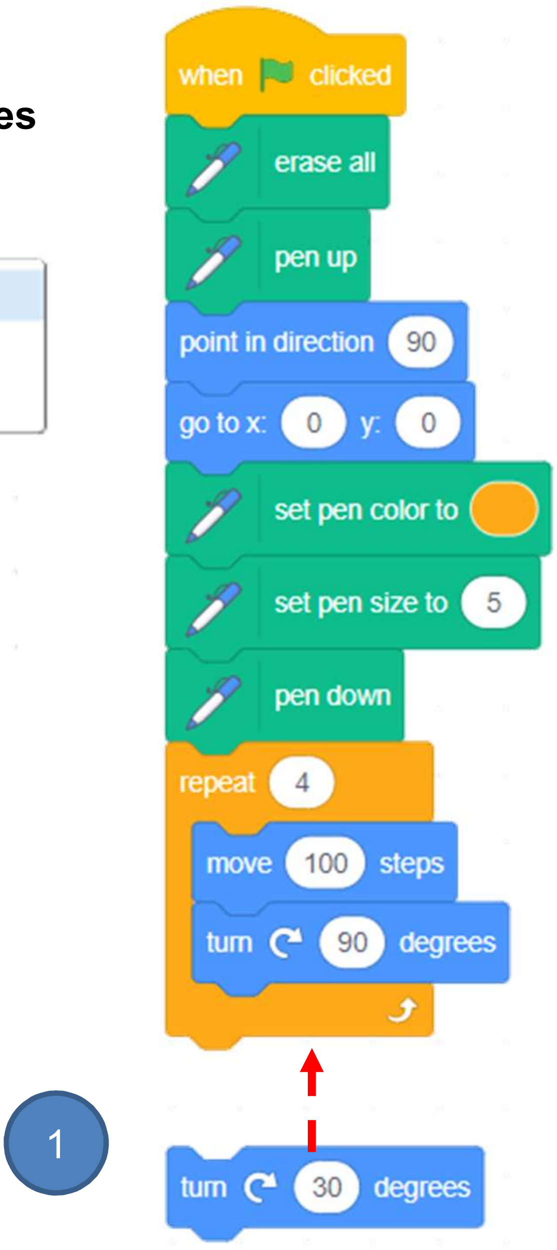
1. Drag the **turn right 15 degrees** block from the **Motion** drawer. Snap it below the **repeat** block and change it to **30** degrees.

2. Duplicate the **repeat** and **turn right 30 degrees** blocks.



3. Snap the duplicate blocks to the **turn right 30 degrees** block.





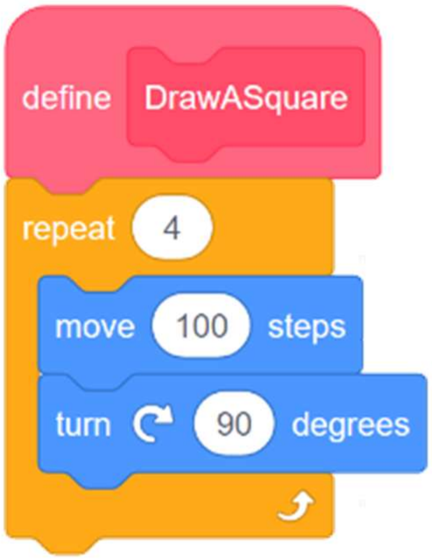
Repeat the steps until you make a snowflake.

Drawing Shapes in Scratch

See Teacher Guide P.17

To Think and To Code: Create “DrawASquare” Block

Drag the blocks that draw a square and snap them to the DrawASquare block.

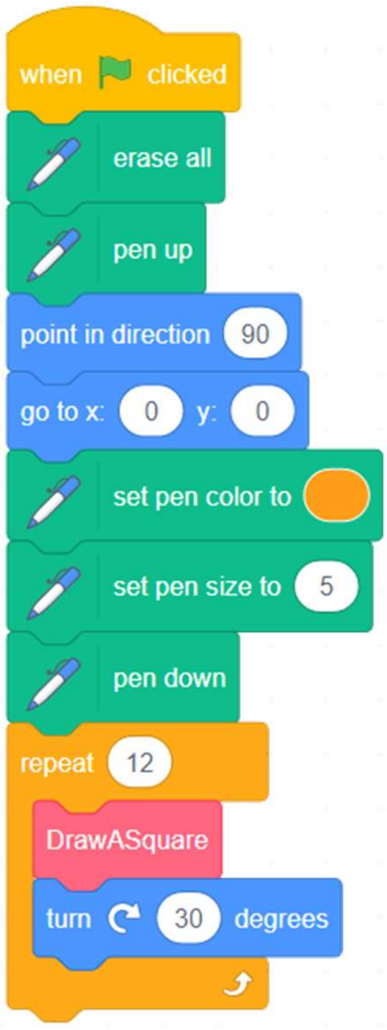


```
define DrawASquare
repeat 4
    move 100 steps
    turn ↻ 90 degrees
```

1. And you need to draw 12 squares to form a snowflake. So, drag a **repeat** block from the **Control** drawer and snap the **DrawASquare** and **turn right 30 degrees** blocks in. Remember to change the number to **12** to draw 12 squares.

2. Before you draw a new square, the sprite needs to turn 30 degrees. So, snap the **turn right 30 degrees** block in the **DrawASquare** block.

3. Drag the **repeat** block set to the end of the when green flag clicked blocks.



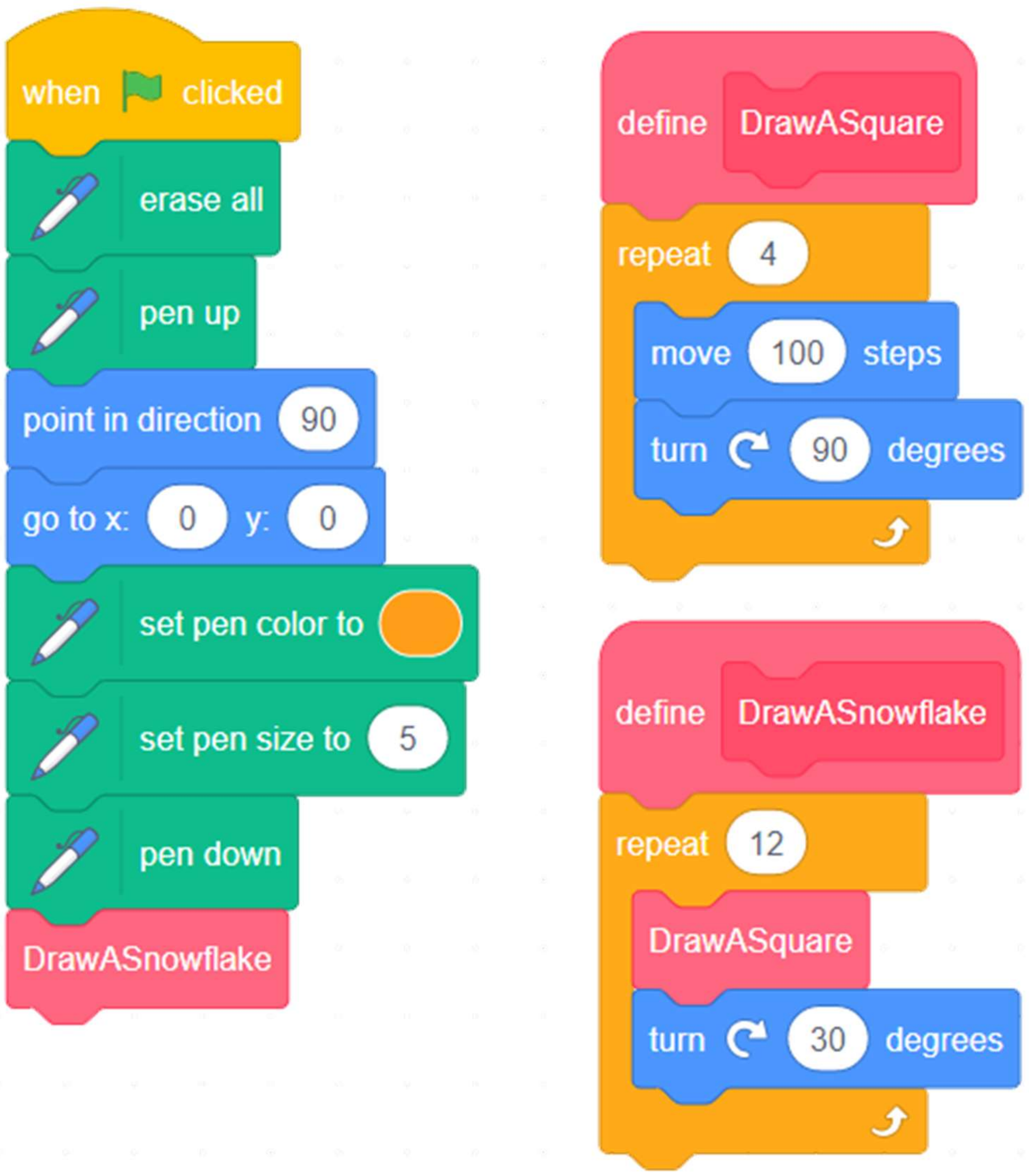
```
when ⚑ clicked
erase all
pen up
point in direction 90
go to x: 0 y: 0
set pen color to ●
set pen size to 5
pen down
repeat 12
    DrawASquare
    turn ↻ 30 degrees
```

Drawing Shapes in Scratch

See Teacher Guide P.18

To Think and To Code: Create “DrawASnowflake” Block

Create and use the “DrawASnowflake” block to make the program code clearer.



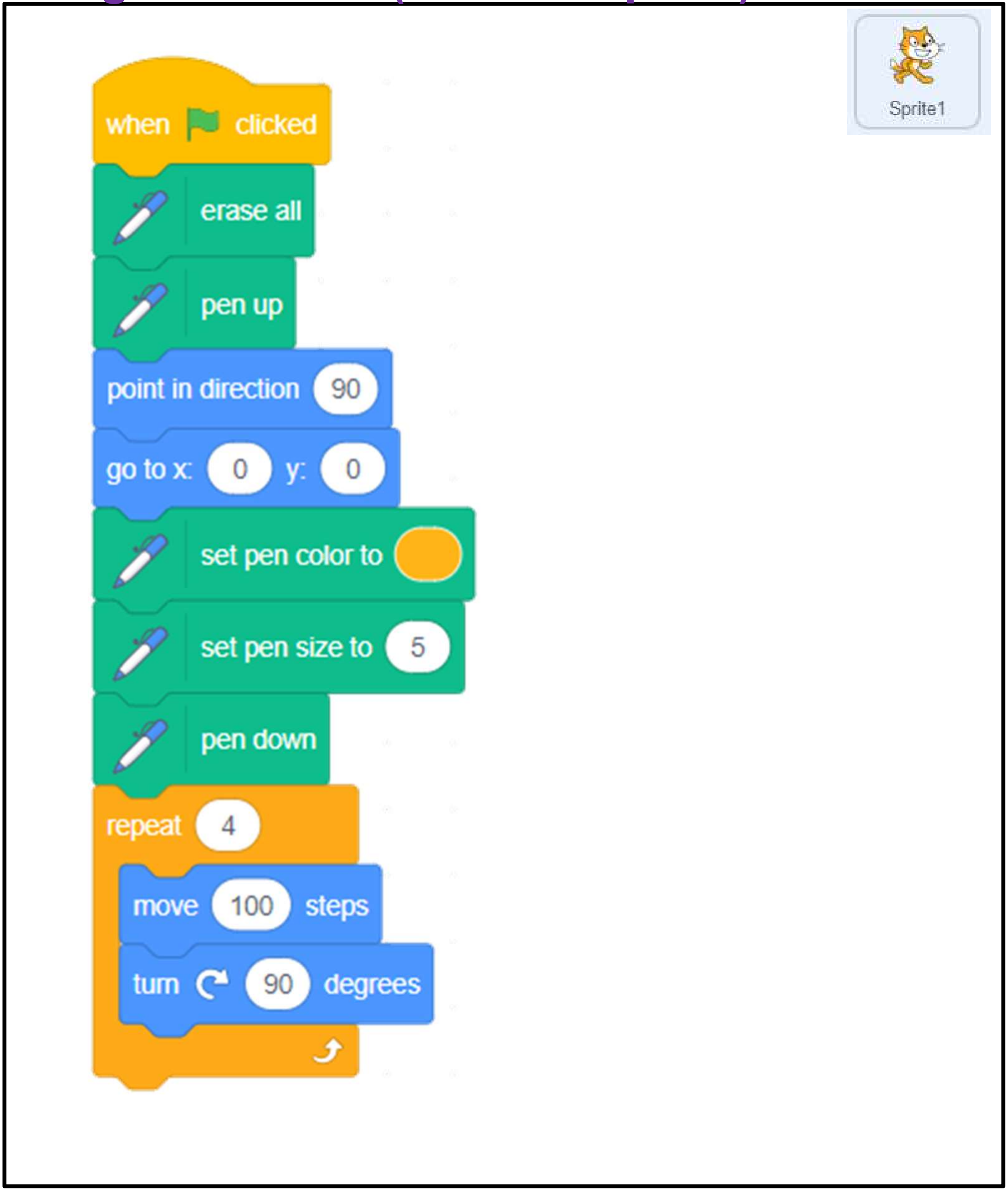
```
when [flag] clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to [ ]
set pen size to (5)
pen down
DrawASnowflake
```

```
define DrawASquare
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
```

```
define DrawASnowflake
repeat (12)
    DrawASquare
    turn ↻ (30) degrees
```

Drawing Shapes in Scratch

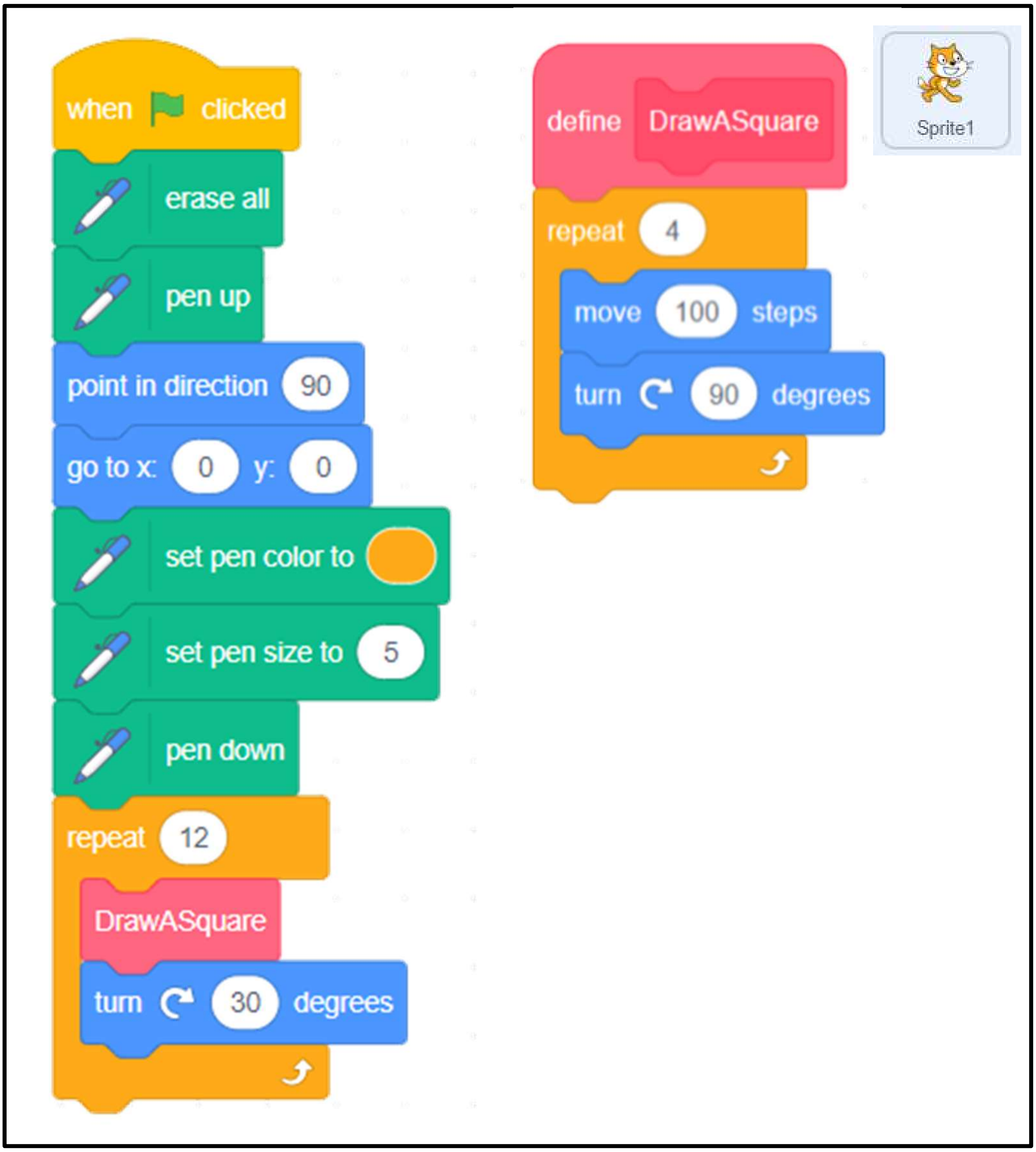
Program Codes (Draw a Square)



```
when 🏴 clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to (🟧)
set pen size to (5)
pen down
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
```

Drawing Shapes in Scratch

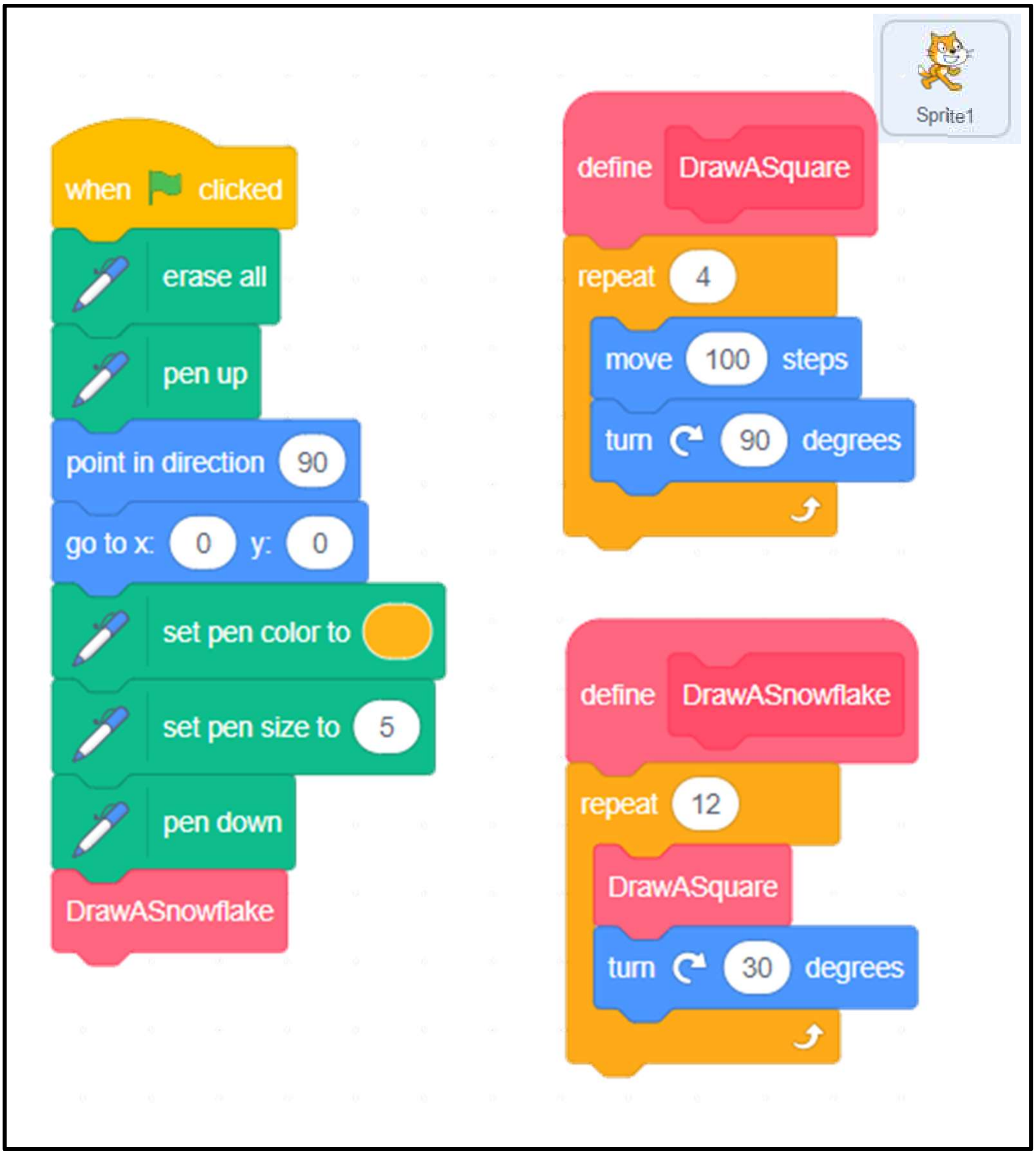
Program Codes (Draw Multiple Squares / a Snowflake – with "DrawASquare" block



```
when 🏴 clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to ( )
set pen size to (5)
pen down
repeat (12)
    DrawASquare
    turn ↻ (30) degrees
```

```
define DrawASquare
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
```

Program Codes (Draw a Snowflake – with "DrawASnowflake" block



```
when 🏳 clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to (●)
set pen size to (5)
pen down
DrawASnowflake
```

```
define DrawASquare
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
```

```
define DrawASnowflake
repeat (12)
    DrawASquare
    turn ↻ (30) degrees
```

Unit 8: Designing Patterns in Scratch (Extension Unit) Teacher Guide

Content

Unit 8: Designing Patterns in Scratch (Extension Unit)
Teaching Plan

Prior Knowledge

Students should have experience with Scratch from the previous units. They should be able to create and make use of "My Block" (know how to name, define and call it), know about the Pen feature and the coordinate as well.

Learning Objectives

1. Create a Scratch project that uses the Pen feature to design line pattern arts through drawing shapes with different size;
2. Develop computational thinking concepts and practices of abstraction and modularization through creating procedures with parameters to draw shapes;
3. Inspire students to be creative in drawing shapes using more advanced coding skills.

Learning Elements

Computational Thinking Concepts and Practices:

Key Learning Elements	Items
Abstraction	Express the algorithm, Modularization
Algorithm	Problem Solving Procedures: Problem identification, Problem analysis, Algorithm design and Programming Basic Programming Constructs: Sequence, Iteration Data Processing: Variable Coding Concept and Practices: Design, Reuse and Remix programs / codes, Testing and Debugging

Coding Skills:

1. Make use of the Pen feature and related code blocks in Scratch;
2. Create and use "My Block" (procedure) with parameters;
3. Reuse and remix programs / codes from drawing a square to multiple squares with different size, combine into pattern;
4. Apply testing and debugging in completing each task of this unit.

Others (including Attitude):

1. Develop interest in programming;
2. Show perseverance and positivity in testing and debugging;
3. Inspire students to be creative and innovative to draw shapes using more advanced coding skills.

Lesson Plan: This unit consists of 2 lessons of 35 minutes.

Lesson 1

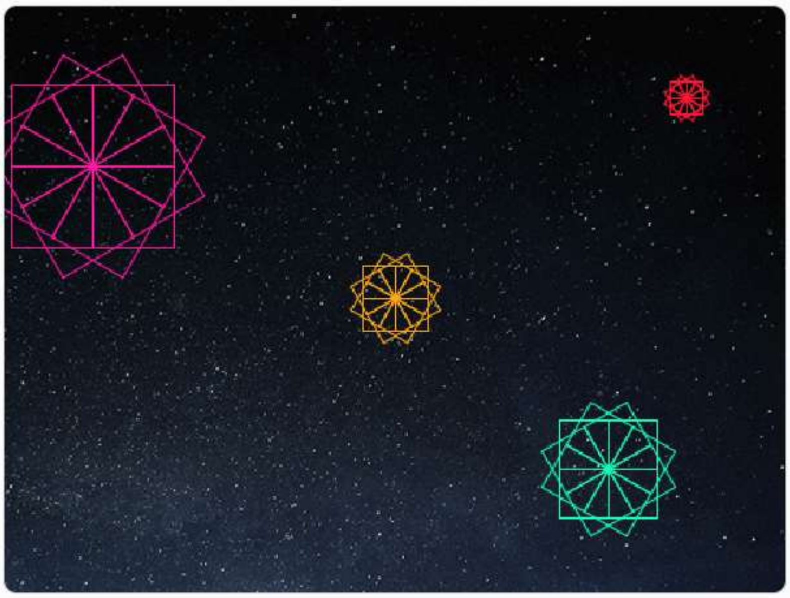
Time	Activity
5 mins	**To Play** 1. Open the Scratch Project: https://scratch.mit.edu/projects/737985288/. 2. Ask students the difference between the patterns of Unit 7 and Unit 8. 3. Tell them that they can click the green flag again and again to see the differences of the patterns.
30 mins	**To Think and To Code** 1. Review the "DrawASquare" block and the concept of simplifying code and calling the blocks to perform a task. 2. Ask students to explain how they made a snowflake shape using a set of rotated squares with the help of mind map. 3. Ask students how they might make several snowflakes in different locations on the stage in a project. 1) Students might suggest copying/pasting code, or using a loop. 2) Ask students if it makes sense to make a custom block to make a snowflake when they want to make several snowflakes, just like they made a block to draw a square. 3) Ask students how they would make a custom block to make a snowflake. 4. Students complete the student guide creating the "DrawASnowflake" block.

Lesson 2

Time	Activity
20 mins	**To Think and To Code** 1. Ask students how they might make bigger snowflakes using bigger squares. Ask students what they would need to change in their code blocks. 2. Changing the "DrawASquare" block so the sprite moves 150 steps, and make a larger square and subsequent snowflake. 3. Ask students to make different sized snowflakes by changing the numbers of steps. 4. Add an input parameter for size to the "DrawASquare" and "DrawASnowflake" blocks.
15 mins	**To Reflect:** **Share the Projects and Provide Constructive Feedback on Program Design** 1. Create a studio and give students the studio URL. Ask students to save and submit their projects to the teacher's Studio. 2. Have one or two students share their project with the whole class. Feedback to be given by peer and teachers. 3. Students should share, think of ways to improve / enhance their program as well as the aesthetic values. **Review of Student Learning** 1. Review on the features of Scratch, and key concepts and practices learnt in the unit. 2. Ask students to complete the review questions, appropriate feedback should be given by teachers.

Designing Patterns in Scratch

In this unit, you will learn how to make more snowflakes in different locations on the stage. Also, you will change the size of your snowflakes in your art project.



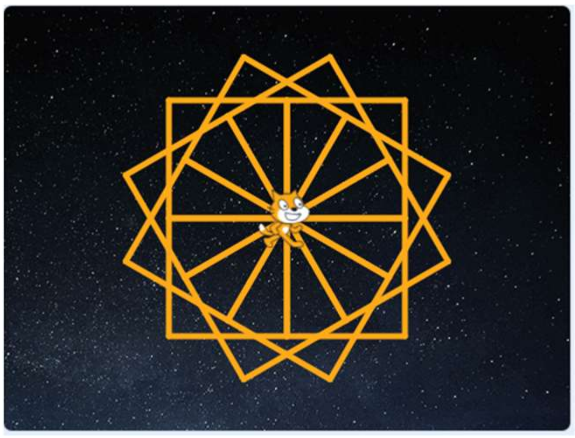
To Play

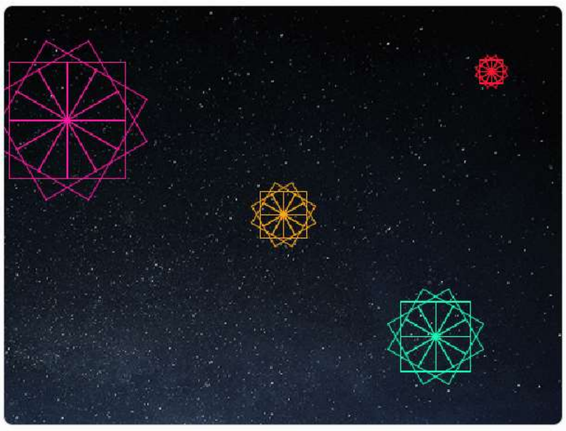
Open the Scratch Project https://scratch.mit.edu/projects/737985288/

Click the green flag and see what will happen.

Click again to see if those snowflakes go to the same way.

Did you see the Scratch cat this time?

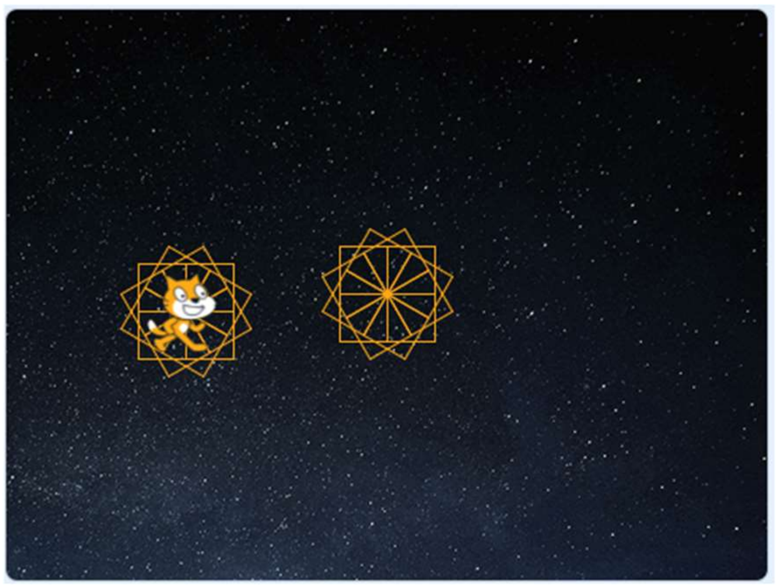




Designing Patterns in Scratch

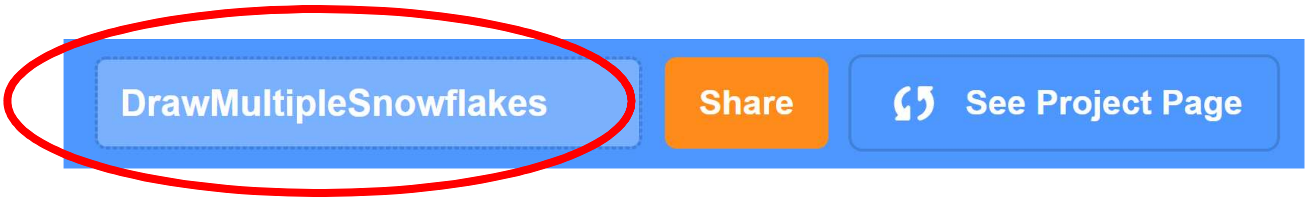
To Think

Do you have any ideas for how to make multiple snowflakes in different locations on the stage?



1. Sign into your account at scratch.mit.edu. Go to "My Stuff" under your name at the right top of the screen and open your Unit 7 "DrawASnowflake" project.

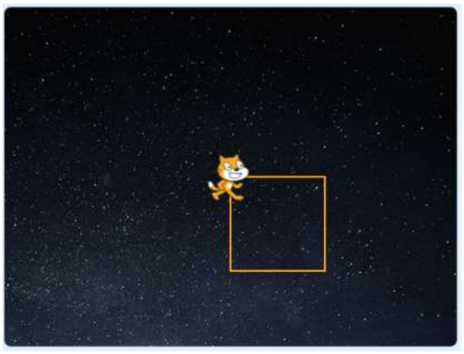
2. Select "Save as a copy" from the "File" menu. Then change the name to "DrawMultipleSnowflakes" and save your project.



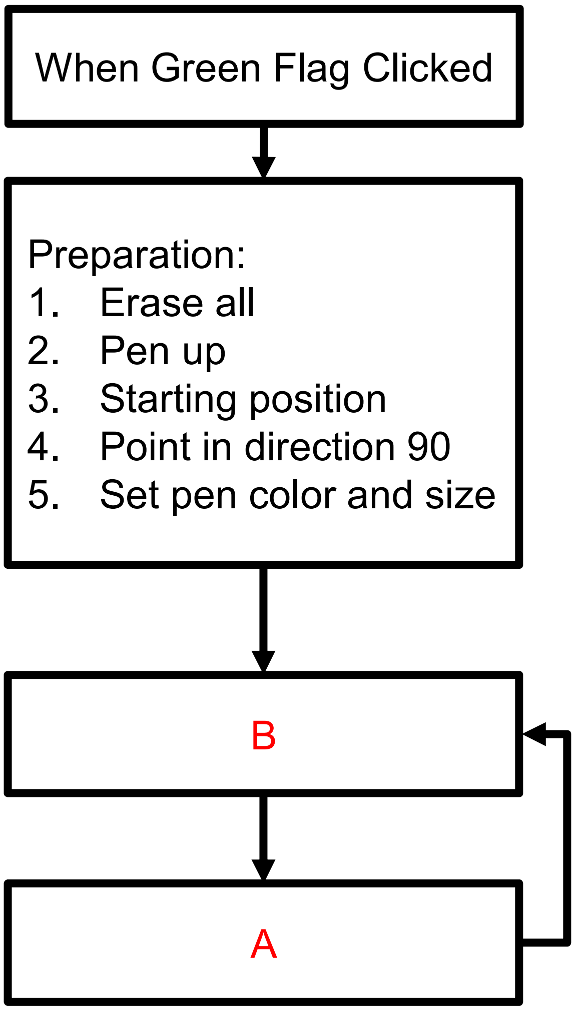
Designing Patterns in Scratch

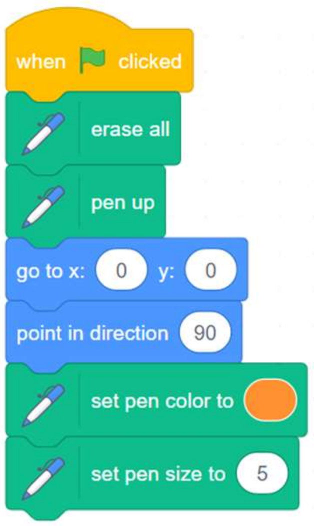
To Think: Draw a Square

Review what you learned in the previous unit, do you remember how to draw a square?



A. Turn 90 degrees	B. Draw a Line





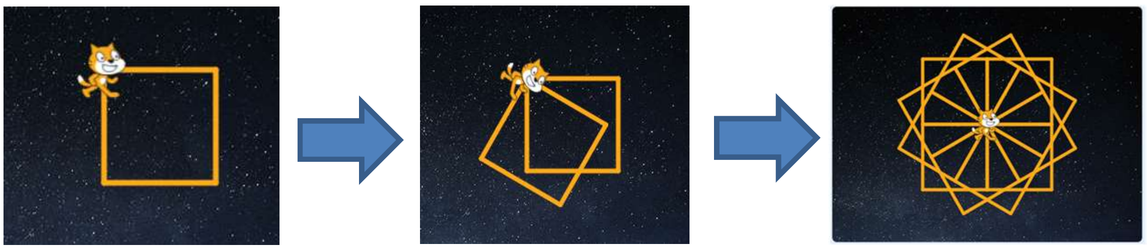
```
when 🏁 clicked
erase all
pen up
go to x: 0 y: 0
point in direction 90
set pen color to ●
set pen size to 5
```

Repeat 4 Times

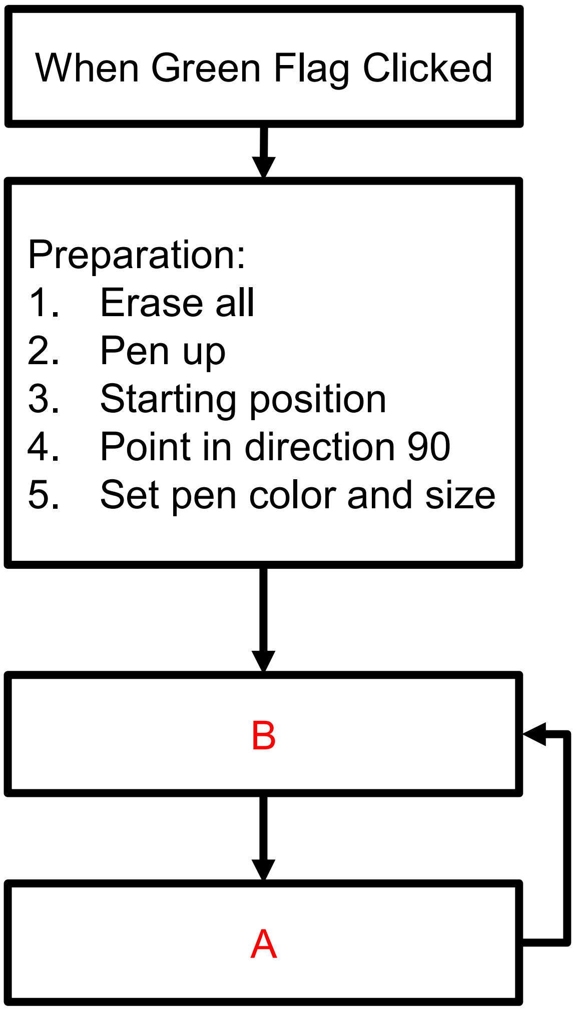
Designing Patterns in Scratch

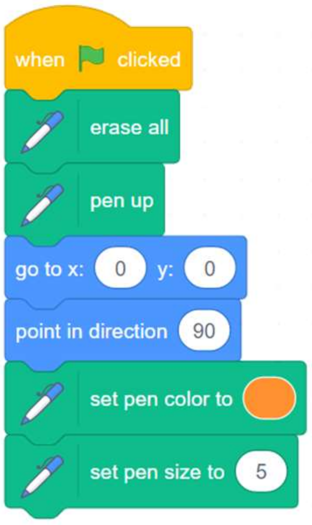
To Think: Draw a Snowflake

Try to extend "Draw a Square" to "Draw a Snowflake".



A. Turn 30 degrees	B. Draw a Square





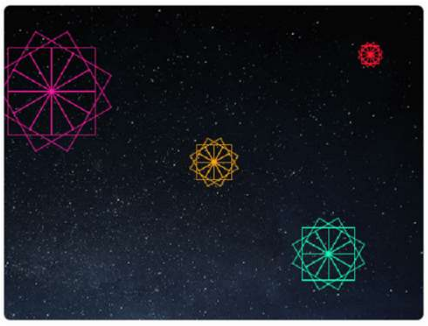
```
when 🏴 clicked
erase all
pen up
go to x: 0 y: 0
point in direction 90
set pen color to ●
set pen size to 5
```

Repeat 12 Times

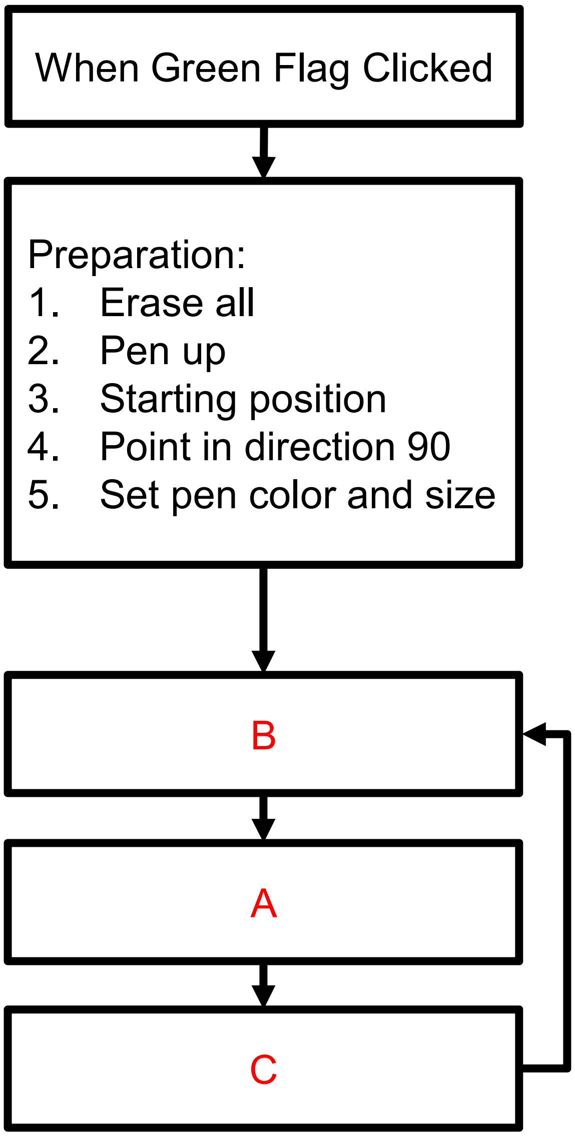
Designing Patterns in Scratch

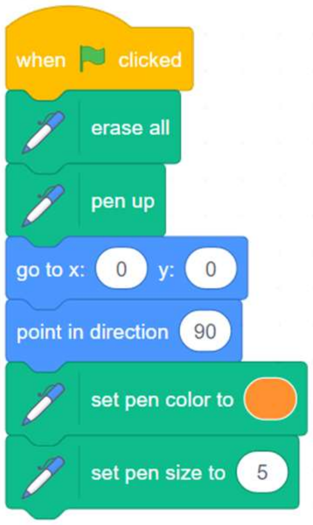
To Think: Draw More Snowflakes

How to draw more snowflakes?



A. Go to Random Position	B. Draw a Snowflake	C. Change Pen Color





```
when 🏴 clicked
erase all
pen up
go to x: 0 y: 0
point in direction 90
set pen color to ●
set pen size to 5
```

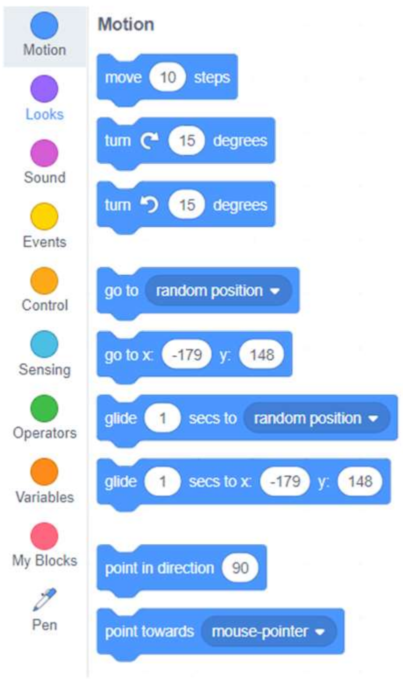
Designing Patterns in Scratch

To Code: Draw More Snowflakes

After drawing one snowflake, the sprite should move to a new position. Code to do it!

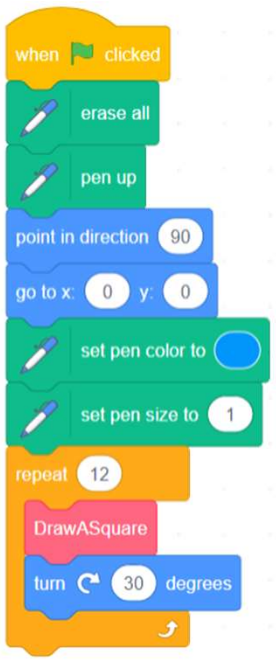
See Appendix P.21

Hint: Check out the "Motion" drawer.



How would you draw a second snowflake once the sprite has moved to a new position?

Hint: Repeat the blocks to add more snowflakes.


```
when green flag clicked
erase all
pen up
point in direction 90
go to x: 0 y: 0
set pen color to
set pen size to 1
repeat 12
    DrawASquare
    turn ↻ 30 degrees
```

Testing and Debugging

Click on the green flag and see if the snowflakes are drawn in different positions. Then look at those blocks. Can you make them shorter?

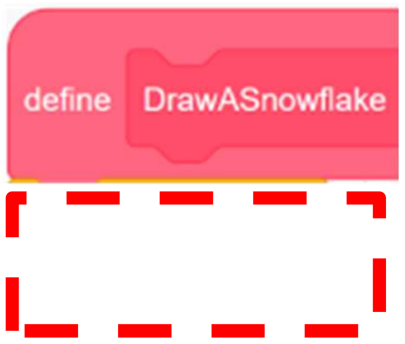
Draw More Snowflakes

Designing Patterns in Scratch

To Think and To Code: Use DrawASnowflake Block

See Appendix P.22

You should have some repeated steps to draw
Review the "DrawASnowFlake" block that we learnt in the previous unit.

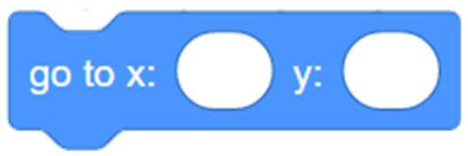


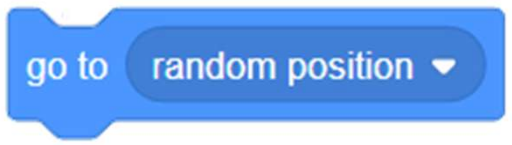
Can you use the "DrawASnowFlake" block to simplify the coding and draw more snowflakes this time?

To draw multiple snowflakes in different locations on the stage, you need to move the sprite to different positions.

Where do you want to go? Certain X / Y or random position?

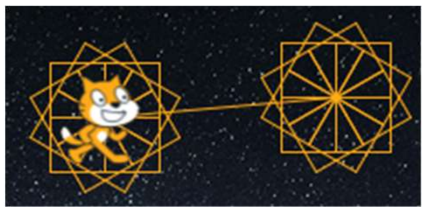
Hint:





Testing and Debugging

Does the simplified blocks do the same thing?
Do you see there is a line between two snowflakes? Why does it happen?



Draw More Snowflakes → Use DrawASnowflake block →

Designing Patterns in Scratch

To Think and To Code: Pen Up & Down

See Appendix P.23

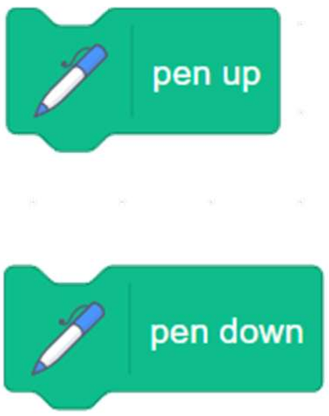
Why is there a line between two snowflakes?

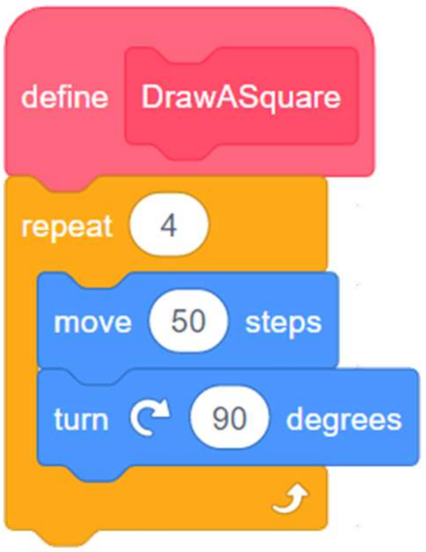
Does the pen always keep "pen down" ?

When drawing a square, keep "pen down"; after drawing, keep "pen up" .

Go to the "Pen" drawer, which block(s) can help you out?
Then update the "DrawASqaure" block.

Hint:




```
define DrawASquare
repeat 4
    move 50 steps
    turn ↻ 90 degrees
```

Testing and Debugging

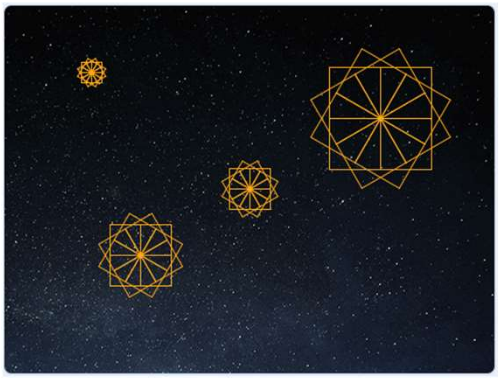
Click to test it! After drawing a snowflake, does the line disappear this time?

Draw More Snowflakes → Use DrawASnowflake block → Pen Up & Down

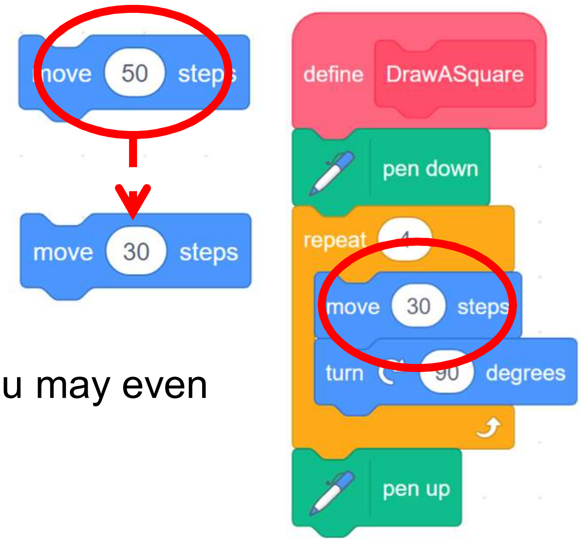
Designing Patterns in Scratch

To Think and To Code: Draw Different Sized Snowflakes

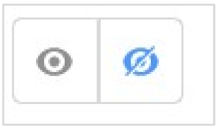
Think about how to make snowflakes of different sizes in different locations?



To make various sizes of snowflakes, we will need to change the size of the squares that make up the snowflakes.



1. In the "define DrawASquare" block", change the number of steps from 50 steps to "move (e.g. 30) steps".
2. Click on the green flag and see what is drawn.
3. If you find the pattern is too small to see, you may even hide the sprite.



Testing and Debugging

If you change the square size by changing the number of "move ( ) steps" block, this will affect the sizes of all snowflakes.

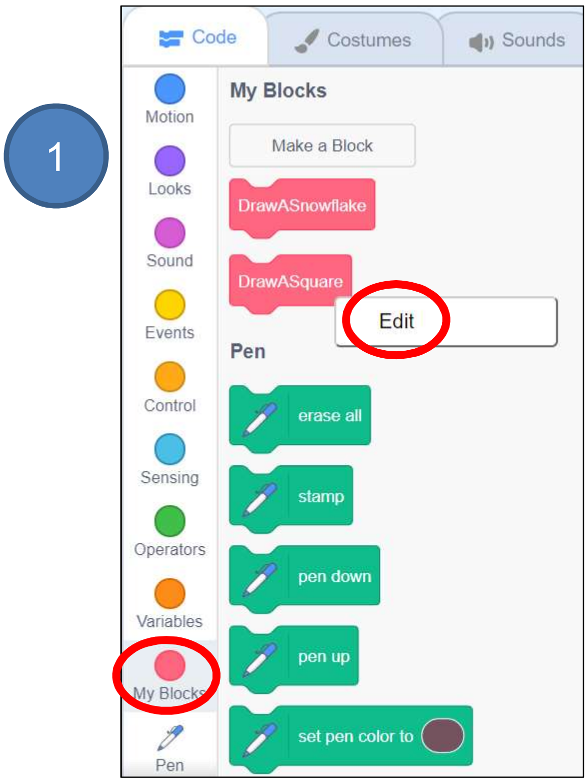
Draw Different Sized Snowflakes

To Think and To Code: Add an Input (number or text) for Custom Blocks

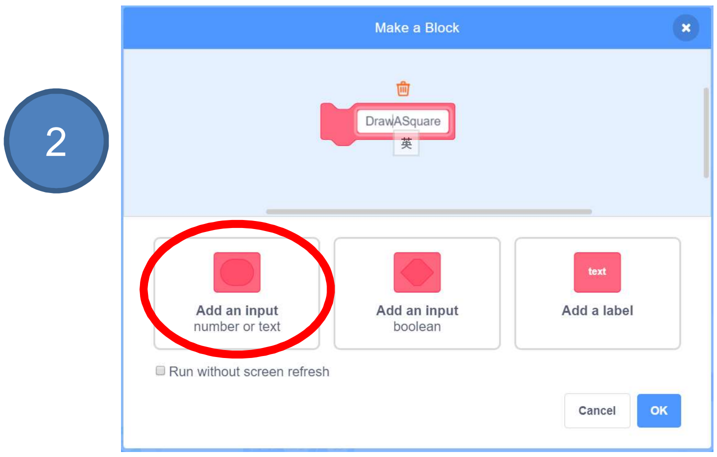
How can you draw each snowflake in a different size? Do you have any ideas? You can create an input parameter for your custom block to create various sizes of squares and snowflakes.

To create an input parameter for your "DrawASquare" block, you need to:

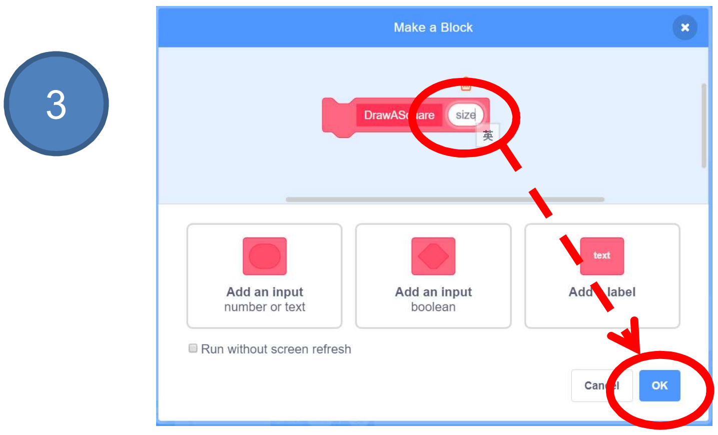
1. Right click the "DrawASquare" block in the "My Blocks" drawer. Click on "Edit".



2. Then click " Add an input ".



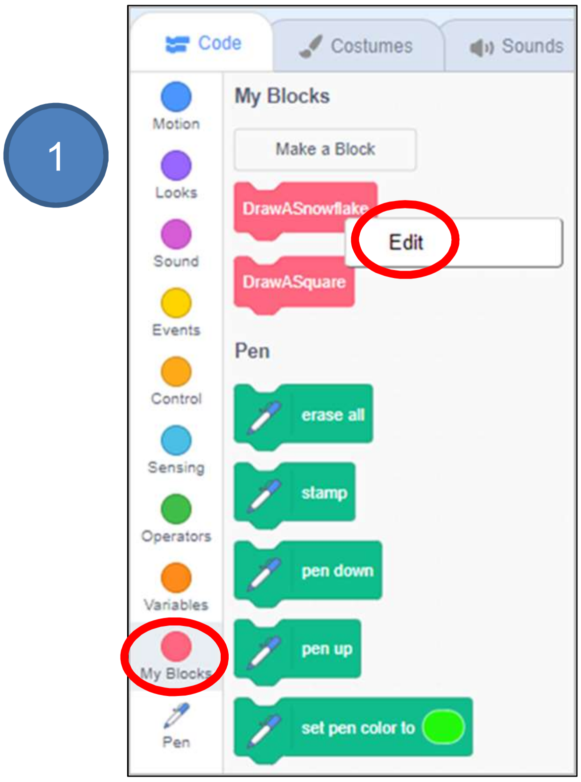
3. Type "size" to name the input parameter, then click the "OK" button to save the "DrawASquare" block.



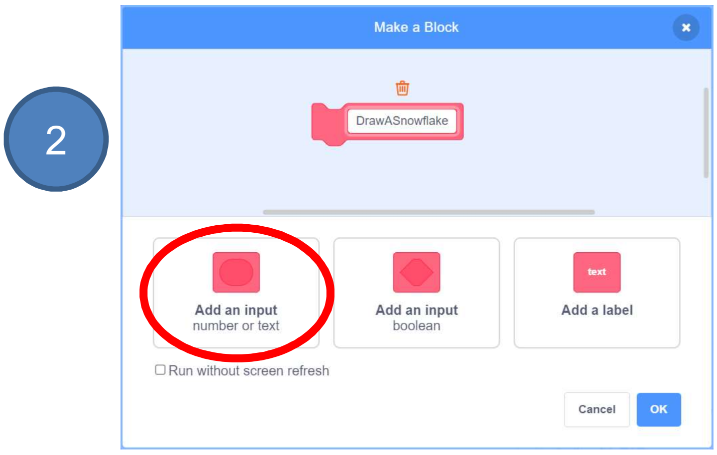
To Think and To Code: Add an Input (number or text) for Custom Blocks

Let's do the same with our "DrawASnowflake" block by adding an input parameter.

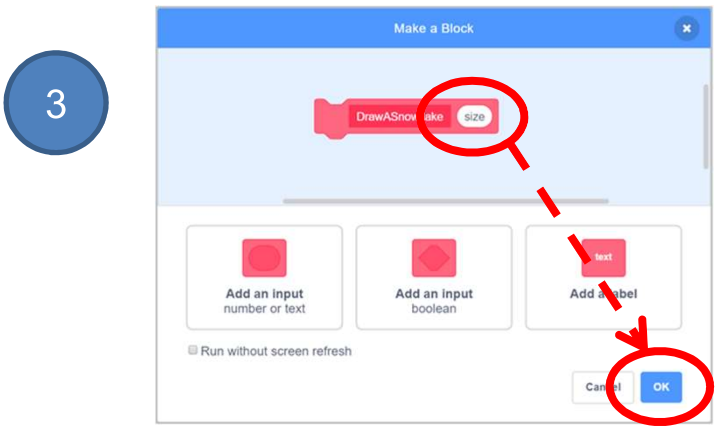
1. Right click the "DrawASnowflake" block in the "My Blocks" drawer. Click on "Edit".



2. Then click "Add an input ".



3. Type "size" to name the input parameter, then click the "OK" button to save the "DrawASnowflake" block.



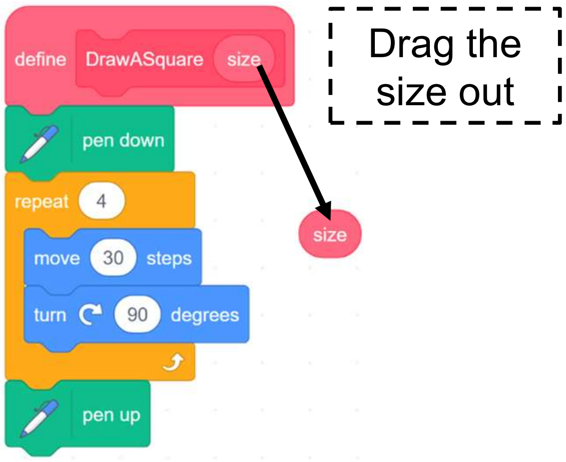
Designing Patterns in Scratch

To Think and To Code: Add an Input (number or text) for Custom Blocks

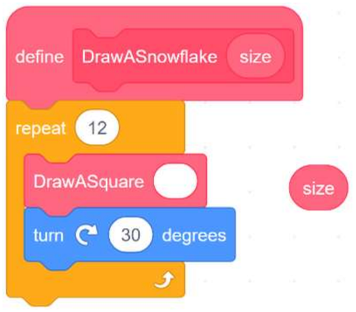
Set the size for “DrawASquare” and “DrawASnowflake” blocks.

See Appendix P.24

1. Drag “ size ” from “ DrawASquare (size) ” out. Where should we put the “ size ” block? Which block should be replaced?

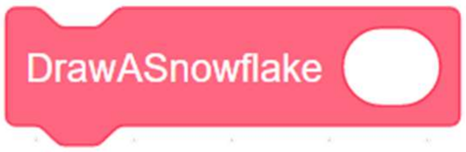


```
define DrawASquare (size)
pen down
repeat (4)
    move (30) steps
    turn ↻ (90) degrees
pen up
```



```
define DrawASnowflake (size)
repeat (12)
    DrawASquare ( )
    turn ↻ (30) degrees
```

2. You should also update the “ DrawASnowflake (size) ” too.
3. Choose different size values for the “ DrawASnowflake ( ) ” block when you use it to draw different sized snowflakes.



Testing and Debugging

Click on the green flag and see what is drawn.
Now, can you add three more snowflakes to your project with different sizes and locations?

Draw Different Sized Snowflakes → Add an Input (number or text) for Custom Blocks

Designing Patterns in Scratch

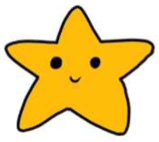
To Reflect: Two Stars and a Wish Worksheet

Name of Project: ___________ Name of Creator: ___________

Please write down two things that you like about this project.

1

"Two Stars and a Wish" is a reflection strategy designed for student feedback as peer- and self-assessment.
Teachers can guide students to give constructive feedback to their peers regarding their Scratch project - two positive (stars) and one hopeful (wish) reflection. Comments can be made on Scratch project's idea, features and aesthetic aspects etc.

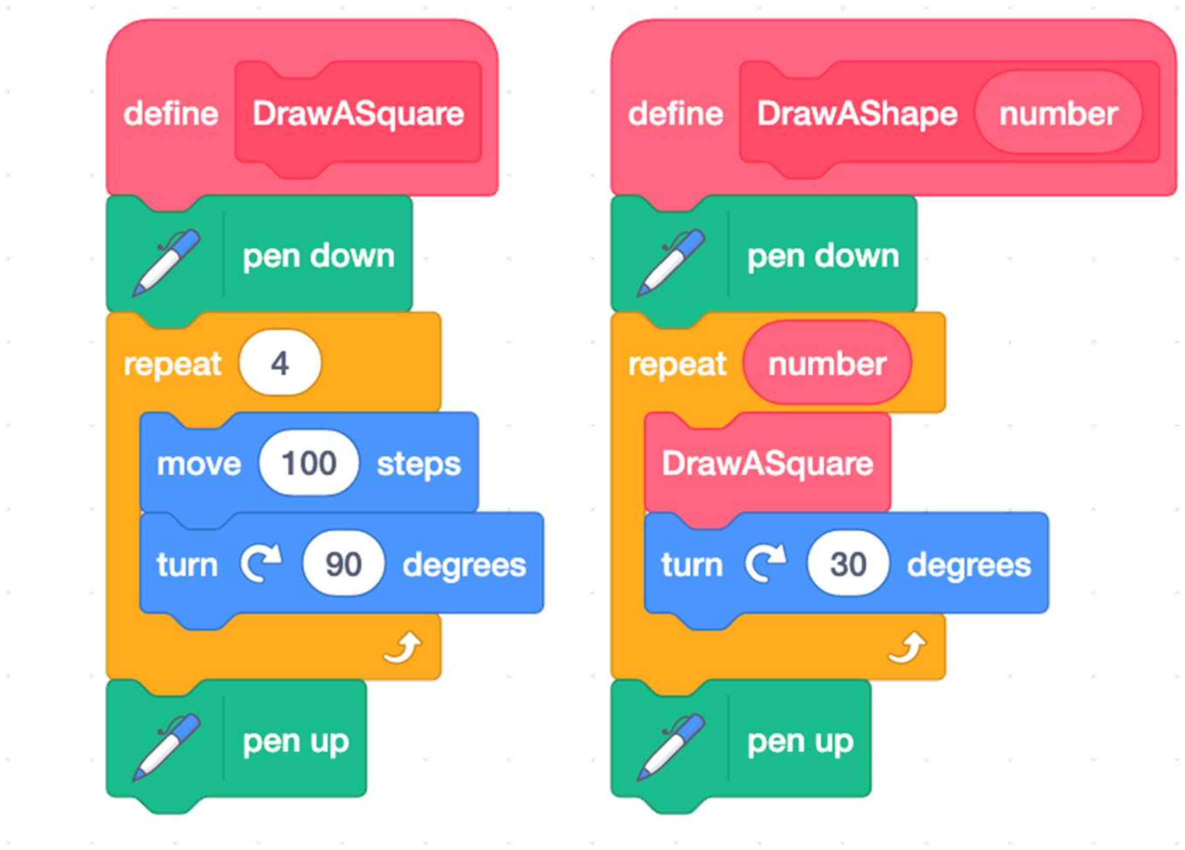

2

What is one thing you would like to add or change to make this project better?

Designing Patterns in Scratch

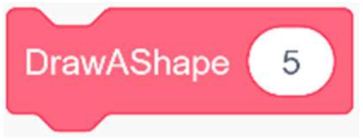
Review Questions

1. A student makes a project with the following custom blocks.



```
define DrawASquare
pen down
repeat 4
    move 100 steps
    turn ↻ 90 degrees
pen up
```

```
define DrawAShape number
pen down
repeat number
    DrawASquare
    turn ↻ 30 degrees
pen up
```

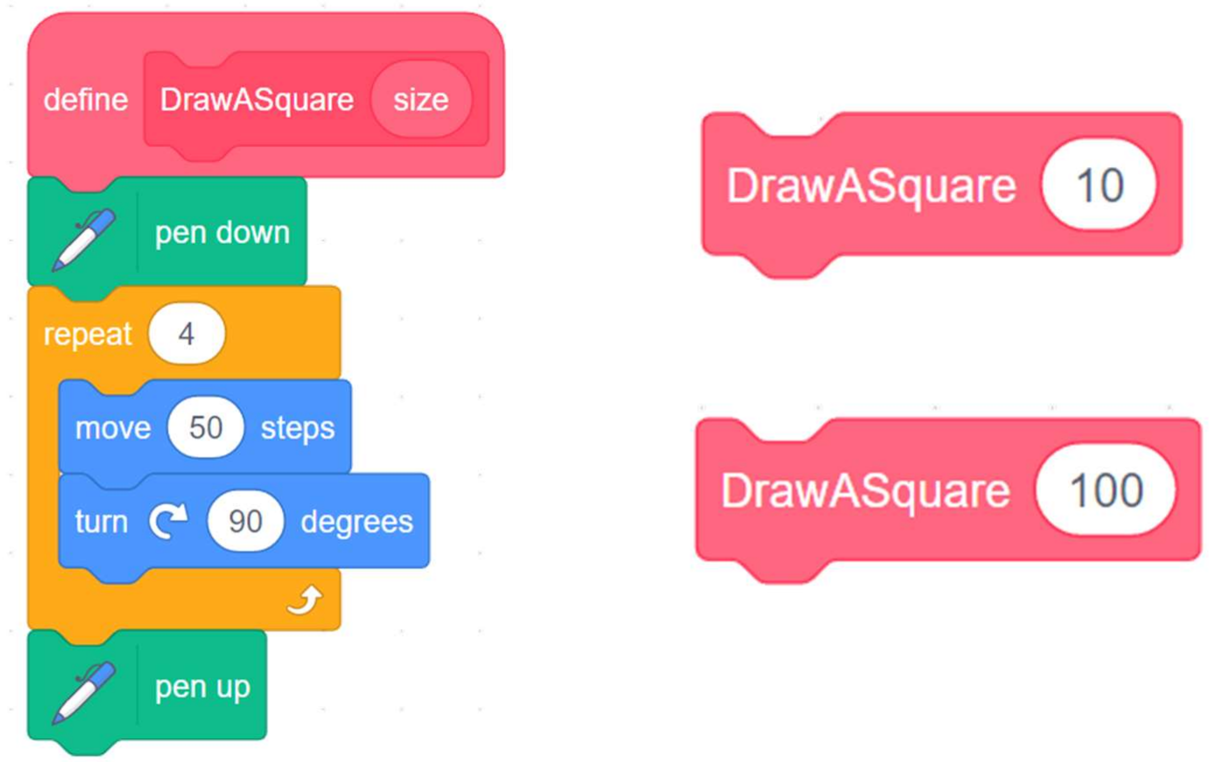
If the student calls  `DrawAShape 5` , what happens?

A. A snowflake with a size of 5 is drawn.
B. 5 snowflakes are drawn.
C. A shape with 5 rotated squares is drawn.
D. Nothing is drawn because the pen is up.

(Answer: C)

Review Questions

2. A student failed to draw squares with different sizes no matter he called DrawASquare (10) or DrawASquare (100). Can you help debugging?


```
define DrawASquare (size)
pen down
repeat (4)
    move (50) steps
    turn ↻ (90) degrees
pen up

DrawASquare (10)

DrawASquare (100)
```

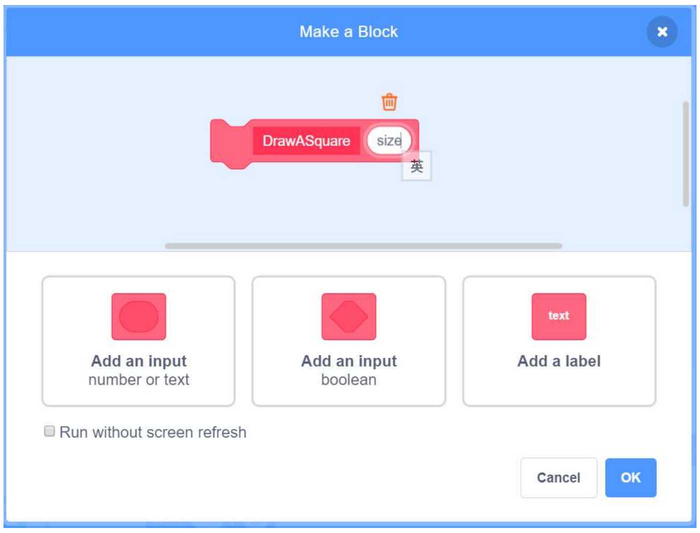
A. Rename the block from "**DrawASqaure**" to "**DrawASnowflake**".
B. Remove the "**size**" parameter inside the "**DrawASqaure**" block.
C. Repeat "**size**" but not "**4**".
D. Replace "**50**" in the "**move 50 steps**" block with the "**size**" parameter.

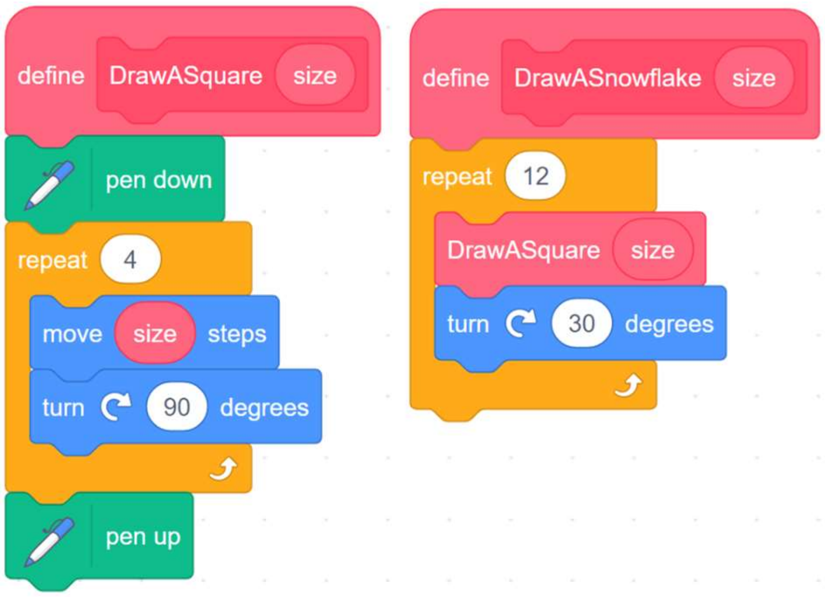
(Answer: D)

Designing Patterns in Scratch

Revision on Key Features

My Blocks with Parameter: With the passing parameter “size”, we can create different sized squares and snowflakes.





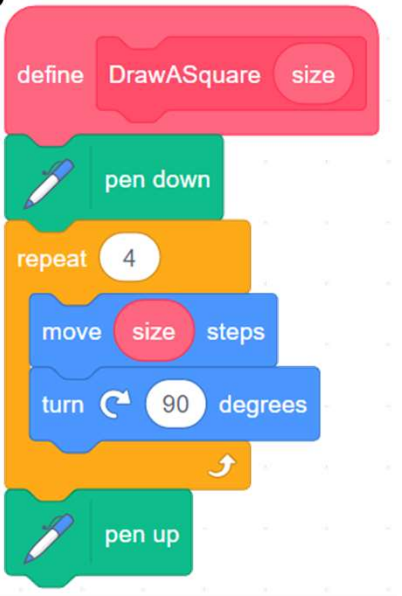
```
define DrawASquare (size)
pen down
repeat (4)
    move (size) steps
    turn ↻ (90) degrees
pen up
```

```
define DrawASnowflake (size)
repeat (12)
    DrawASquare (size)
    turn ↻ (30) degrees
```

Designing Patterns in Scratch

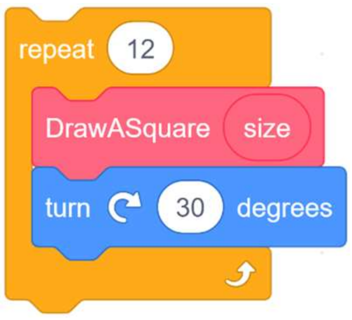
Revision on Key Concepts & Practices

Sequence: It is the order in which the programming statements are executed. A wrong order would lead to incorrect programming results.

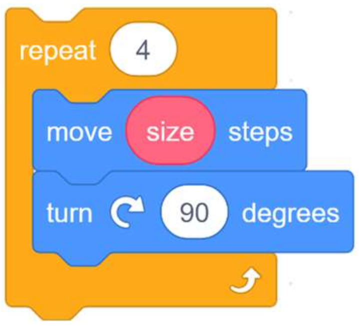


```
define DrawASquare (size)
pen down
repeat (4)
    move (size) steps
    turn ↻ (90) degrees
↺
pen up
```

Iteration: Iteration is repeating a process in order to generate a sequence of outcomes. “Repeat___” block can trigger iteration in Scratch.



```
repeat (12)
    DrawASquare (size)
    turn ↻ (30) degrees
↺
```



```
repeat (4)
    move (size) steps
    turn ↻ (90) degrees
↺
```

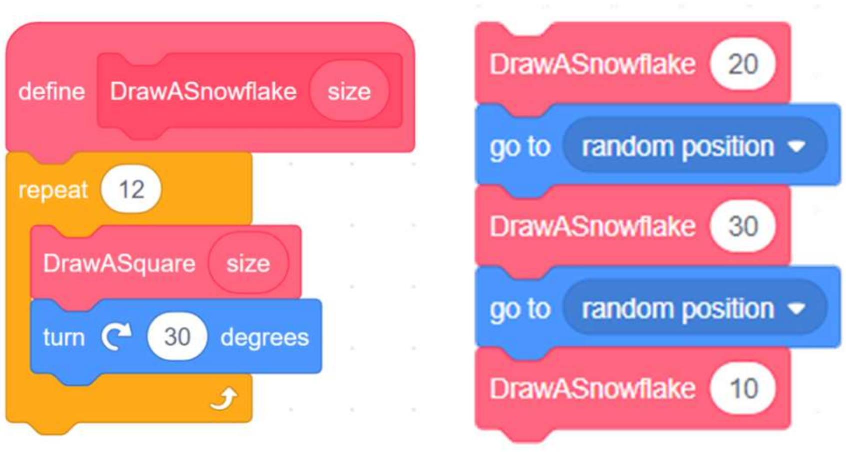
Designing Patterns in Scratch

Revision on Key Concepts & Practices

Abstraction and modularization: In computer programming, abstraction is the process of identifying key information that is relevant in a given context and forgetting those details in that context. For example, the name the custom block (procedure) “DrawASnowflake” capture its abstract key meaning of what to do later in the module.

In computer programming, modularization is the process of organizing code for the task it executes and always try to keep the code reusable. For example, we can use the “DrawASnowflake” module repeatedly to draw snowflakes.

Also, we add a variable, "size" which we call a "parameter", to the custom blocks. This allows us to slightly change the modules. With the parameter “size”, we can create different sized squares and snowflakes.



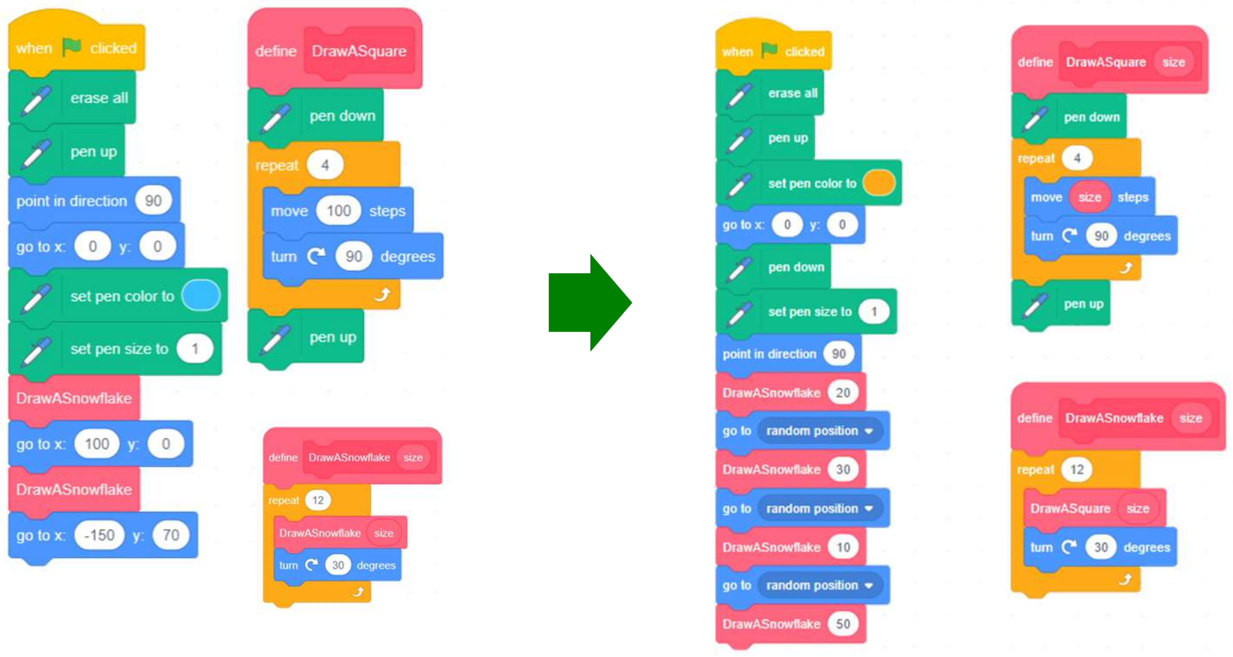
```
define DrawASnowflake size
repeat 12
    DrawASquare size
    turn ↻ 30 degrees
```

```
DrawASnowflake 20
go to random position
DrawASnowflake 30
go to random position
DrawASnowflake 10
```

Designing Patterns in Scratch

Revision on Key Concepts & Practices

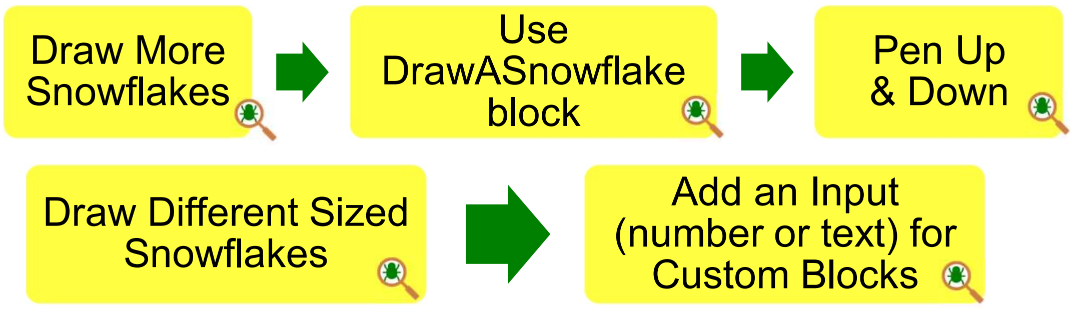
Being incremental and iterative: to work out a sub-task as an iteration, try it out, then work out another sub-task in one more iteration until the whole programming task is completed.



```
when [flag] clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to ( )
set pen size to (1)
DrawASnowflake
go to x: (100) y: (0)
DrawASnowflake
go to x: (-150) y: (70)

define DrawASquare
pen down
repeat (4)
    move (100) steps
    turn ↻ (90) degrees
pen up

define DrawASnowflake (size)
repeat (12)
    DrawASnowflake (size)
    turn ↻ (30) degrees
```

```
when [flag] clicked
erase all
pen up
set pen color to ( )
go to x: (0) y: (0)
pen down
set pen size to (1)
point in direction (90)
DrawASnowflake (20)
go to (random position ▾)
DrawASnowflake (30)
go to (random position ▾)
DrawASnowflake (10)
go to (random position ▾)
DrawASnowflake (50)

define DrawASquare (size)
pen down
repeat (4)
    move (size) steps
    turn ↻ (90) degrees
pen up

define DrawASnowflake (size)
repeat (12)
    DrawASquare (size)
    turn ↻ (30) degrees
```

Testing and Debugging: Testing a computer program is the process of checking if it can produce results as designed. Debugging a computer program is the process of finding out ways to revise the program so that the bugs can be removed.


Draw More Snowflakes
Use DrawASnowflake block
Pen Up & Down
Draw Different Sized Snowflakes
Add an Input (number or text) for Custom Blocks

Appendix

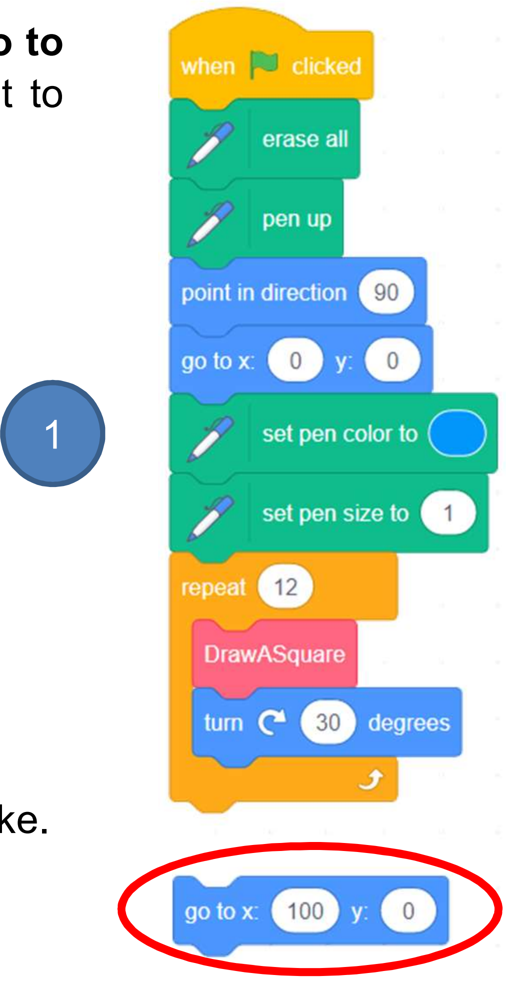
Operation Manual

Designing Patterns in Scratch

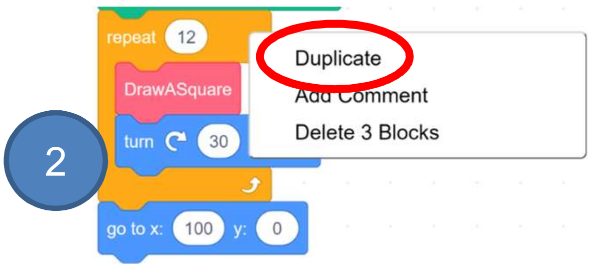
See Teacher Guide P.6

To Code: Draw More Snowflakes

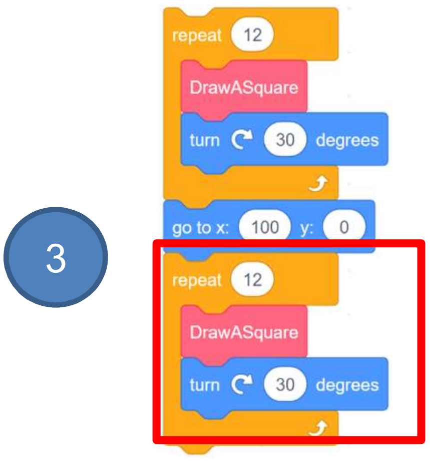
1. To move the sprite to a new position, drag a **go to x:0 y:0** block from the **Motion** drawer. Snap it to the above blocks and change **x to 100.**



2. Duplicate the **repeat** block that draws a snowflake.



3. Snap the duplicated **repeat** block to the **go to x:100 y:0** block.

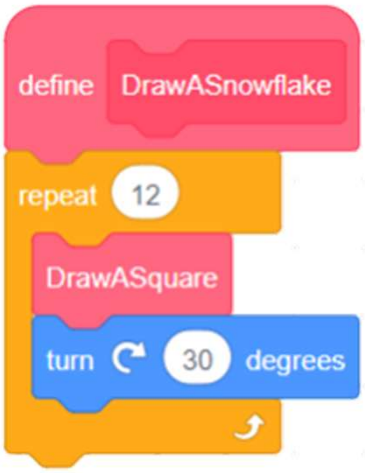


Designing Patterns in Scratch

To Think and To Code: Use DrawASnowflake block

See Teacher Guide P.7

Revision: Drag the blocks that draw a snowflake and snap them to the **define DrawASnowflake** block.

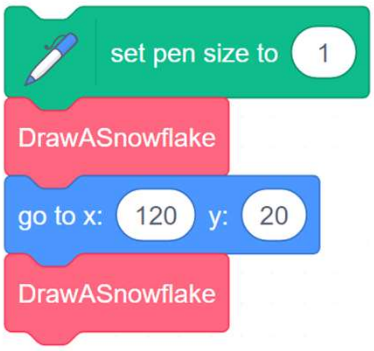


To draw multiple snowflakes in different locations on the stage, you need to move the sprite to different positions.

1. Drag a **go to x: y:** block from the **Motion** drawer. Snap it to the existing blocks.

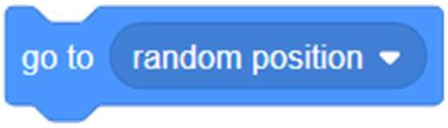
2. Change the x and y values of the **go to x: y:** block to different values of your choosing to change the position of the snowflake.

3. Drag a **DrawASnowflake** block and snap it to the **go to x: y:** block.



If you want to add more snowflakes, just repeat step 1 to 3.

You may also try **go to random position** instead of to certain position.



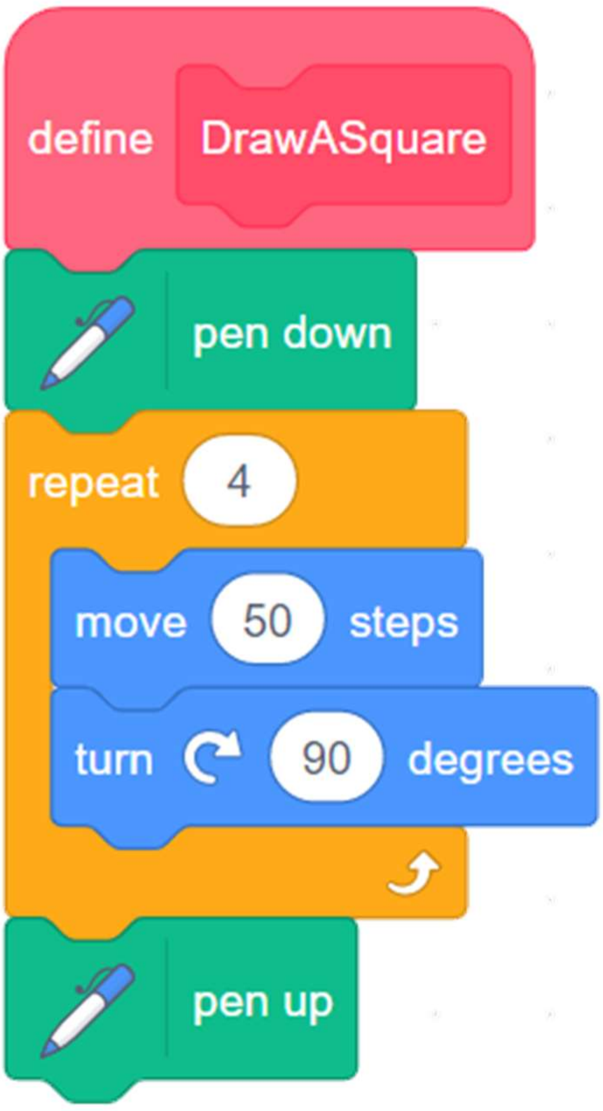
Designing Patterns in Scratch

To Think and To Code: Pen Up & Down

See Teacher Guide P.8

When excuting the "DrawASquare" block, to draw a square, keep “pen down”; after drawing, keep “pen up” .

If you think the pattern is too big/small , you may also update the move (e.g. 50) steps.



```
define DrawASquare
pen down
repeat (4)
    move (50) steps
    turn ↻ (90) degrees
pen up
```

Designing Patterns in Scratch

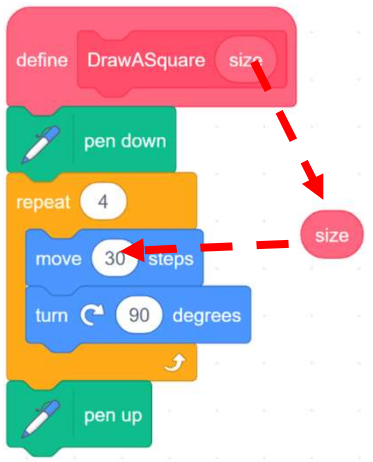
To Think and To Code: Add an Input (number or text) for Custom Blocks

See Teacher Guide P.12

Set the size for **DrawASquare** and **DrawASnowflake** blocks.

1. Drag **size** from **DrawASquare (size)** to replace **30** in the **move 30 steps** block in the **DrawASquare (size)** custom block.

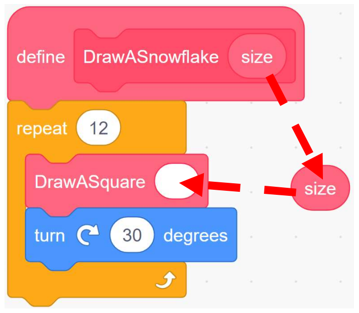
1



```
define DrawASquare (size)
pen down
repeat (4)
    move (30) steps
    turn ↻ (90) degrees
pen up
```

2. Drag **size** from **DrawASnowflake (size)** into the empty slot ( ) of the **DrawASquare ( )** block in the **DrawASnowflake (size)** custom block.

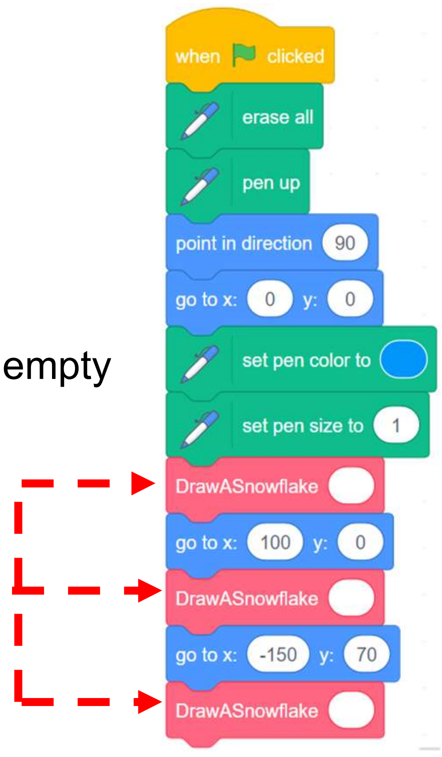
2



```
define DrawASnowflake (size)
repeat (12)
    DrawASquare ( )
    turn ↻ (30) degrees
```

3. Choose three different size values for each empty slot ( ) of the **DrawASnowflake ( )** blocks.

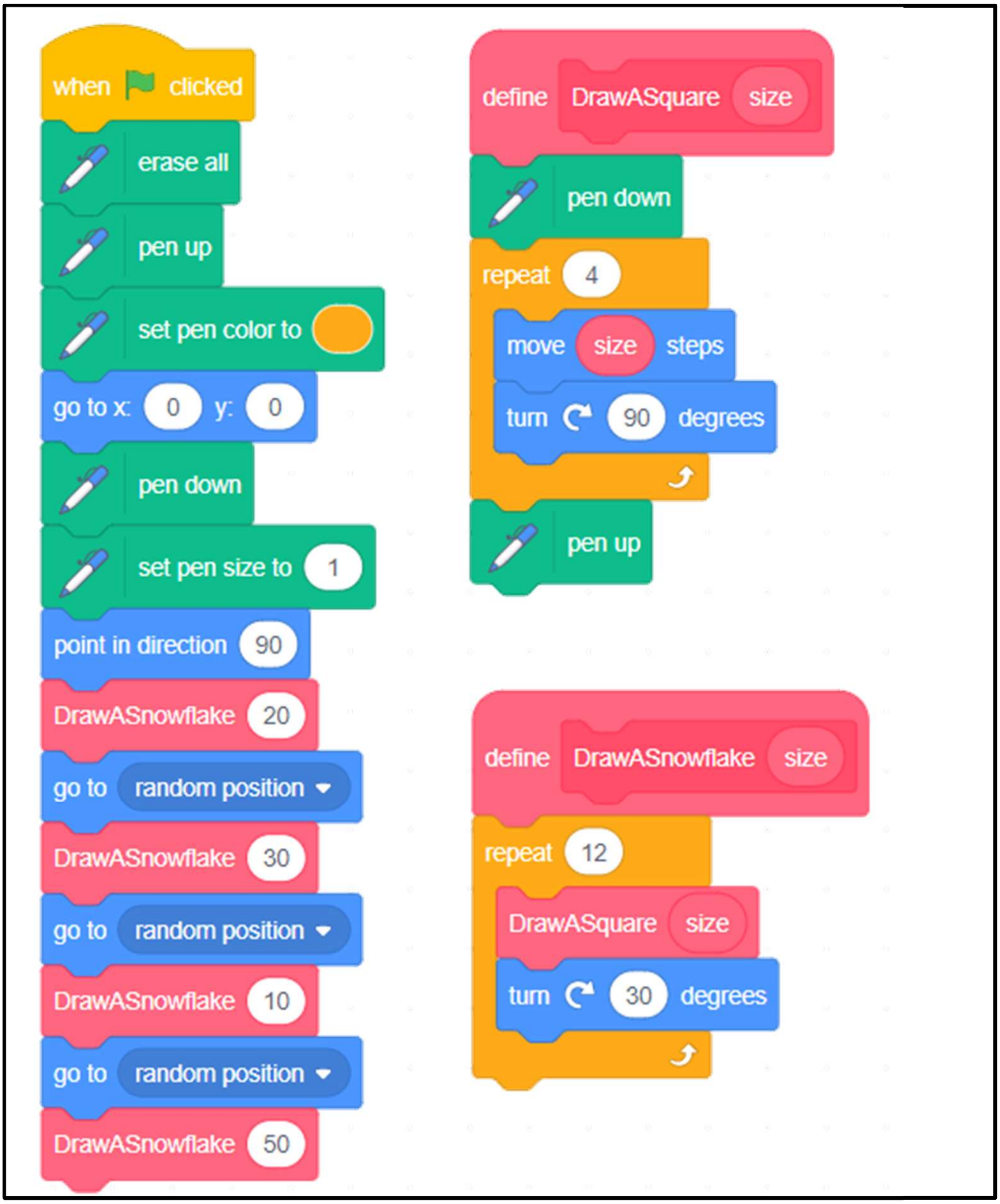
3



```
when ⚑ clicked
erase all
pen up
point in direction (90)
go to x: (0) y: (0)
set pen color to ( )
set pen size to (1)
DrawASnowflake ( )
go to x: (100) y: (0)
DrawASnowflake ( )
go to x: (-150) y: (70)
DrawASnowflake ( )
```

Designing Patterns in Scratch

Program Codes



```
when 🏁 clicked
erase all
pen up
set pen color to
go to x: 0 y: 0
pen down
set pen size to 1
point in direction 90
DrawASnowflake 20
go to random position
DrawASnowflake 30
go to random position
DrawASnowflake 10
go to random position
DrawASnowflake 50
```

```
define DrawASquare size
pen down
repeat 4
    move size steps
    turn ↻ 90 degrees
pen up
```

```
define DrawASnowflake size
repeat 12
    DrawASquare size
    turn ↻ 30 degrees
```

Scratch Final Project
Teacher Guide

Content

Scratch Final Project
Teaching Plan

Prior Knowledge

Students have completed Units 1 to 8 in Scratch, so they will have a solid understanding of the computational thinking concepts and practices and coding skills needed to fulfil the project requirements.

Learning Objectives / Expected Learning Outcomes

1. Identify a problem and suggest solution to solve it by using Scratch;
2. Design a Scratch project to solve the problem;
3. Apply computational thinking concepts and practices to complete the Scratch project;
4. Foster students' creativity by creating their own coding artefacts and understand that coding can be a fun and social experience through sharing their projects with others.

Learning Elements

Computational Thinking Concepts and Practices:

Key Learning Elements	Items
Abstraction	Express the algorithm / Modularization
Algorithm	Problem Solving Procedures: Problem identification, Problem analysis, Algorithm design and Programming Basic Programming Constructs: Sequence, Branching / Selection, Iteration Data Processing: Variable Coding Concept and Practices: Design, Reuse and Remix programs / codes, Testing and Debugging
Interacting with Physical Objects	Micro:bit / video sensor

Coding Skills:

Students should have a solid understanding of all coding skills covered in Unit 1-8.

Others (including Attitude):

1. Develop interest in programming;
2. Show perseverance and positivity in testing and debugging;
3. Inspire students to be creative and innovative to enhance their final projects.

Lesson Plan: This unit consists of 8 lessons of 35 minutes.

Teacher Preparation for the Lesson: Tell students that they will design their own Scratch project to solve a problem they face in daily life in this unit. Ask them to think about what problem they would like to solve and how to solve it by using Scratch. The project can be a game or a story.

Lesson 1 and 2

Time	Activity
5 mins	**Introduction** 1. Explain to students that they will work in pairs to design a Scratch project to showcase their Scratch knowledge. 2. Explain to them they will have 8 lessons for this project, including a group presentation at the end.
10 mins	**Review Basic Programming Constructs, Key Concepts and Practices, and Scratch Features** 1. Review the basic programming constructs, key concepts and practices, and the Scratch features they have learnt and used in the previous units. Encourage them to use various blocks in their project. The details can be found in Appendix.
20 mins	**Problem Identification** 1. Ask students to pair up and discuss with their partner and identify a problem and think of a way to solve it using Scratch under the guidance of the worksheet 2. Guide the students to think about different issues in their daily life (e.g. helping to improve the living of the people in need, recycling for protecting our environment, etc.) 3. The project can be a game or a story 4. Some examples are presented for teachers' reference 5. Ask students to jot down their ideas on the worksheet.
20 mins	**Design Your Scratch Story or Game** 1. Ask students to use the mind map to list the features of their story, or use the flow diagram to design their game 2. Draft the storyboard in *"Design Your Scratch Story: Storyboard"* worksheet if they want to create a story 3. Ask students to fill out *"To-Do Checklist"* for today's lesson based on their list of steps required to make their project

	4. Remind students that they can always add or revise the plan and checklist anytime during coding.
10 mins	**Coding Activity** Students start working on their Scratch projects with their partners.
5 mins	**Wrap-up** 1. Check in with students to make sure they have made progress on their projects 2. Ask students to update their *To-Do Checklist* with what has been accomplished, and any issues they encountered. Make plans for next lesson.

Lesson 3 and 4

Time	Activity
5 mins	**Introduction to Lesson** 1. Ask students to review their *To-Do Checklist*. Look over what they have accomplished 2. Add steps for what they hope to accomplish in today's lesson.
60 mins	**Coding Activity** Students implement Scratch projects with their partners.
5 mins	**Wrap-up** 1. Check in with students to make sure they have made progress on their projects 2. Ask students to update their *To-Do Checklist* with what has been accomplished, and any issues they encountered. Make plans for next lesson 3. Remind students that they will show their work to another group in the next lesson.

Lesson 5 and 6

Time	Activity
5 mins	**Introduction to Lesson** 1. Explain to the students that they will work with another group today to provide feedback on each other's projects. 2. Remind students to be constructive in their feedback.
20 mins	**Peer Feedback** 1. Each group will meet with another group and show them their project so far. Each group should also explain what remaining steps they have left to complete their project. 2. Each group will provide feedback to another group on their project using the *Peer Assessment Worksheet.*
15 mins	**Design Revision** Students update their design based on feedback from their peers.
25 mins	**Coding Activity** Students continue to implement the remaining parts of their Scratch projects with their partners.
5 mins	**Wrap-up** 1. Check in with students to make sure they have made progress on their projects. 2. Ask students to update their *To-Do Checklist* with what has been accomplished, and any issues they encountered. Make plans for next lesson. 3. Explain to students that they will have to plan and give a 3-minute presentation of their project in the next lesson.

Lesson 7 and 8

Time	Activity
5 mins	**Introduction to Lesson** 1. Explain to students that they will present their projects in this lesson. They should reserve some time to plan for their presentation. 2. Ask students to look at their *To-Do Checklist*. Look over what they have accomplished and what is remaining and what they can accomplish in the last 10 minutes.
10 mins	**Coding Activity** Students complete Scratch projects with their partners.
10 mins	**Presentation Planning** 1. Ask students to plan a 3-minute presentation for each group on their Scratch Project. 2. As a first step, ask students to upload their projects to the teacher's Final Project Studio so they can run the project. Include the following information. 1) Yourself and your groupmates; 2) The theme / goal of the project (the problem you identified and solved); 3) What you are most proud of your project; 4) Difficulties you overcome.
40 mins	**Presentation** Students present their projects to the class. Remind students to pay attention to the other groups' presentation and provide constructive feedback by using *Group Presentation Peer Assessment Worksheet*.
5 mins	**Wrap-up** 1. Ask students to review the *Group Presentation Peer Assessment Worksheet* completed by their classmates. 2. Complete the *"Self-Assessment"* for self-reflection.

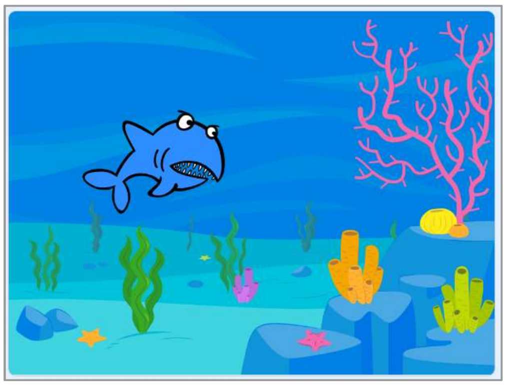
Scratch Final Project

HERE IS THE CHANCE TO SHOW YOUR CODING SKILLS AND DIGITAL CREATIVITY!

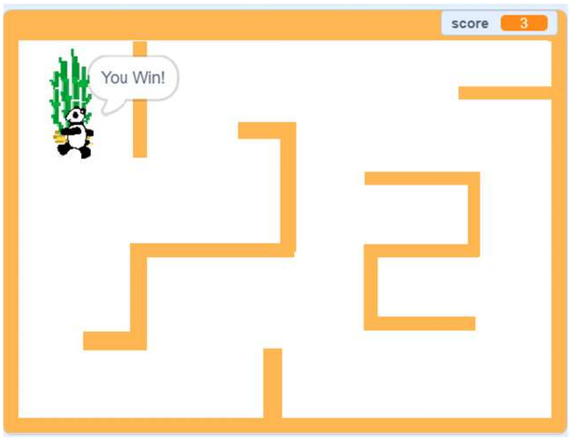
Teacher should try to recall students' memory on what computational thinking concepts and skills, as well as the Scratch features, they have learnt in the previous units, and encourage them to apply them to complete their Scratch Final Project.

- ❑ In this project, you will design and build a Scratch project.
- ❑ Before you start, let's play all Scratch projects again in the previous units and review what features you have learnt so far!

See Appendix P.25-26



Do you still remember the fish exploring under the sea, the panda walking in a maze, and Gobo having a picnic with his friend?



Scratch Final Project Design Worksheet

- ❑ Work in pairs to design and build a Scratch project. Set a theme of the project by identifying a problem or an issue in your daily life and solve it by developing a Scratch programme! (e.g. helping to improve the living of the people in need, recycling for protecting our environment etc.)

- ❑ It can be a game or a story. Think about what kind of project you would like to make. Remember that you will have 8 lessons for this project. (Teachers can adjust the number of lesson to cater student's ability. The design of this project is for reference.)

Try to guide students to think of a problem they want to solve. Hints can be given to those students who do not have any ideas.

To Think…

- ❑ Think about the kind of project you and your groupmates would like to make.
- ❑ Which topic would you like to choose?
- ❑ Would you like to take one of your previous Scratch projects and build something based on that?

Problem Identification

Discuss with your groupmates, think about the Scratch project's theme and design. Use this worksheet to jot down your ideas.

Example (1) for teacher's reference.

1. Does anyone around you is facing problem? Who is she/he?

Grandma

Grandma would like to learn more about technology but she does not have chances to explore.

2. What problem is he/she facing?

Grandma would like to learn more about technology but she does not have chances to explore.

3. How would you like to use Scratch to solve his/her problem? Describe your idea.

Story:
I can use Scratch to design a story. I want to make a story for my grandma. It is about traveling around the world with my grandma and my father in the metaverse.

Problem Identification

Discuss with your groupmates, think about the Scratch project's theme and design. Use this worksheet to jot down your ideas.

Example (2) for teacher's reference.

1. Does anyone around you is facing problem? Who is she/he?

Grandma

Grandma is getting old. She needs to keep her mind active.

2. What problem is he/she facing?

Grandma is getting old. She needs to keep her mind active.

3. How would you like to use Scratch to solve his/her problem? Describe your idea.

Game:
I want to design a game on Scratch for my grandma.

The game will ask my grandma to use the keyboard (up/down/left/right keys) to control the cat to eat fruits.

Problem Identification

Below are some examples for reference.

What would you do if you face the following situations?

Grandma would like to learn more about technology but she does not have chances to explore.

Nowadays, some students are not getting enough exercise and even facing obesity, can you create a Scratch project to provide some tips on how to keep healthy?

Your classmates feel stressed before examination. Can you tell her a joke in Scratch and make her feel relaxed?

Your little brother is not familiar to the types of recycling. Try to create a recycling game for him?

Your little sister loves playing Mathematics games. How about creating a quiz by using Scratch?

Your friend loves music! Can you think of creating a music box that can play different music?

Design your Scratch Story

How would you like to use Scratch to make a story? Use the mind map to list the features of your story.

Sample for teacher's reference.

Theme:
Travel with My Grandma

Sprite

No. of Costume
3 Costume

Blocks
(Circle the used blocks)
Wait / Move / Say / Sound / Show / Hide / Others: Broadcast

Sprite

No. of Costume
3 Costume

Blocks
(Circle the used blocks)
Wait / Move / Say / Sound / Show / Hide / Others: Broadcast

Backdrop

Home
Arctic,
Desert
City

Blocks
(Circle the used blocks)
Wait / Move / Say / Sound / Show / Hide / Others: ___

Other sprite

No. of Costume
2 Costume

Blocks
(Circle the used blocks)
Wait / Move / Say / Sound / Show / Hide / Others: Broadcast, Go to, Glide

Design your Scratch Story: Storyboard

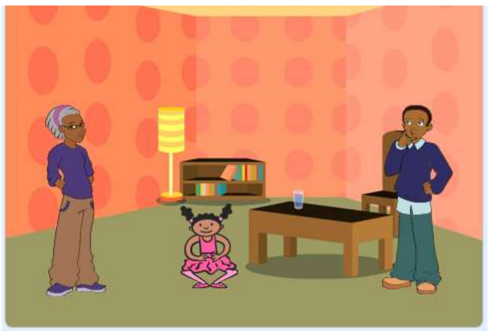
Try to draw pictures to represent the sequence of your story in the boxes below. Put number 1, 2, 3 and 4 in the circles.

Try to think about the theme and write down your ideas below:

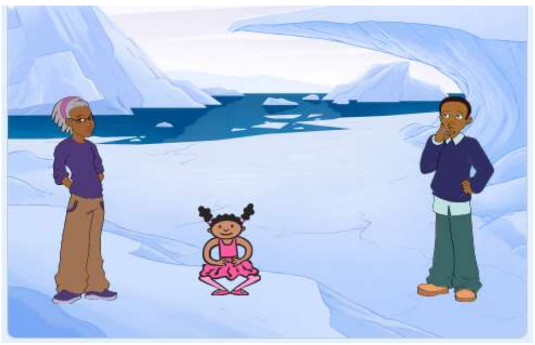
1. Describe the costumes/ motion of your sprites in the story.
2. Introduce the design of using different backdrops

Sample (1) “Travel with My Grandma” (Outstanding)
For teacher’s reference：https://scratch.mit.edu/projects/777194022

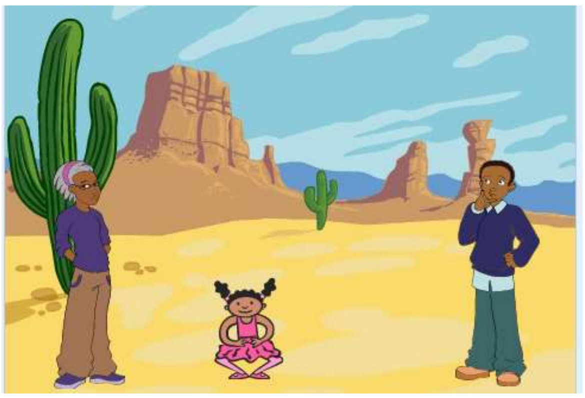
1 Backdrop of the first scene is our home, three sprites (Grandma, father and I) were ready to travel around the world at home. Sound played at the same time.



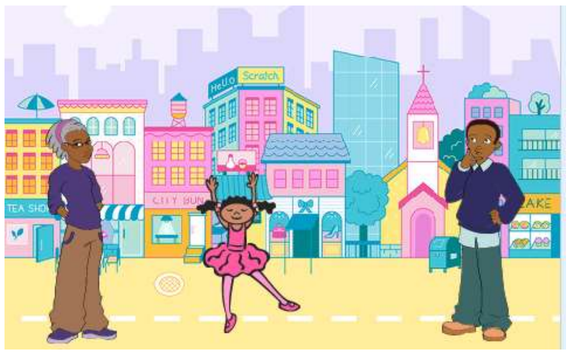
2 We went to North Pole as our fist destination. Backdrop was changed to Artic. Grandma said she could feel the cold! Sound played at the same time.



3 Then we went to desert with a desert backdrop. Grandma said she felt much warmer. Sound played at the same time.



4 Finally we went back to our Smart City. Backdrop was switched to city. I changed my costume too. Sound played at the same time.



Design your Scratch Story: Storyboard

Try to draw pictures to represent the sequence of your story in the boxes below. Put number 1, 2, 3 and 4 in the circles.

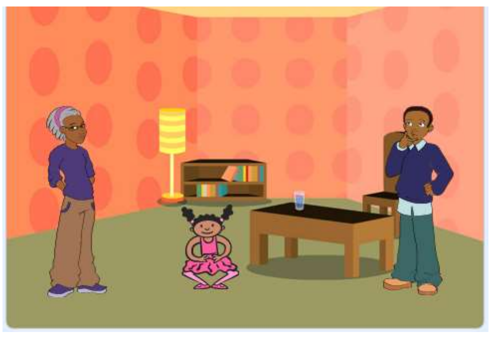
Try to think about the theme and write down your ideas below:

1. Describe the costumes/ motion of your sprites in the story.
2. Introduce the design of using different backdrops

Sample (1) "Travel with My Grandma" (Good)
For teacher's reference: https://scratch.mit.edu/projects/822757916

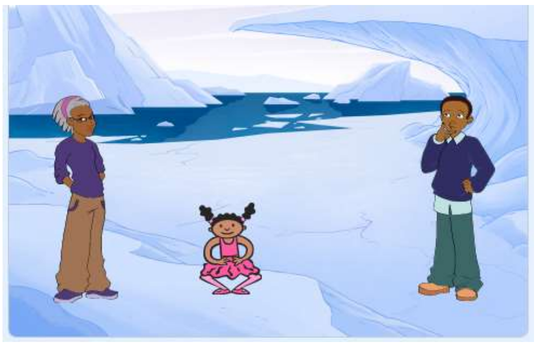
1

Backdrop of the first scene is our home, three sprites (Grandma, father and I) were ready to travel around the world at home.



2

We went to North Pole as our fist destination. Backdrop was changed to Artic. Grandma said she could feel the cold! Sound played at the same time.



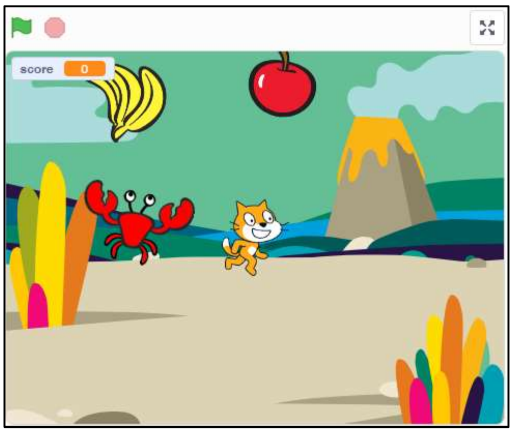
3

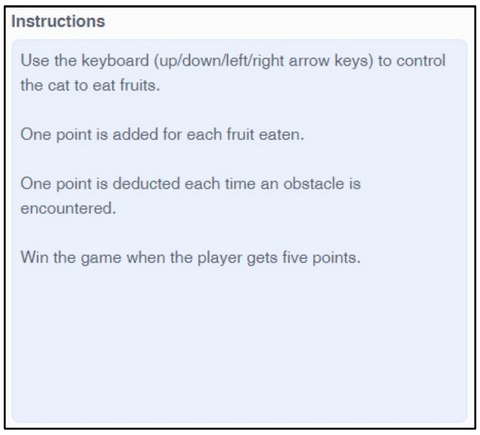
4

Design your Scratch Game

How would you like to use Scratch to make a game? Use the flow diagram to design your game.

Sample (2) "Fruit Game (Outstanding) for teacher's reference:
https://scratch.mit.edu/projects/859723964/



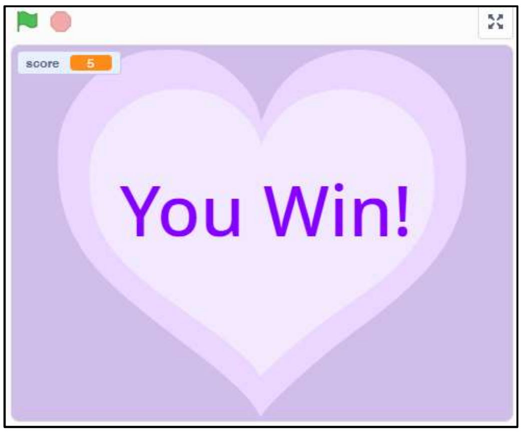

Instructions

Use the keyboard (up/down/left/right arrow keys) to control the cat to eat fruits.

One point is added for each fruit eaten.

One point is deducted each time an obstacle is encountered.

Win the game when the player gets five points.



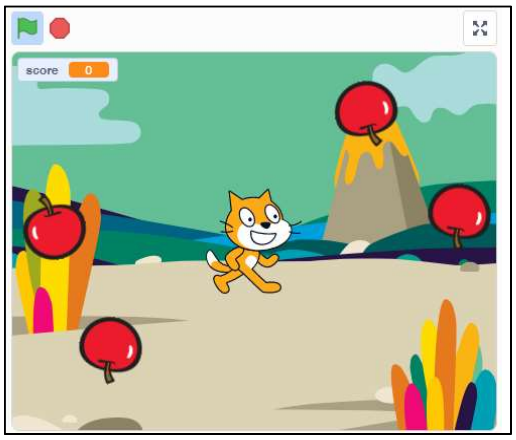
Game Design:

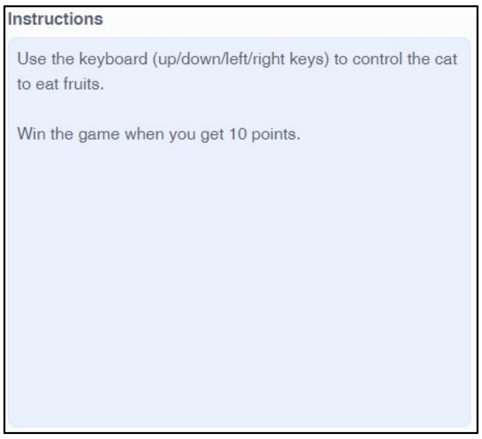
Sprite	Apple, Banana , Bat, Crab, Cat
Motion	up/down/left/right arrow keys, Touch, Score
Design / Rules of the Game	Drag apple or cake sprite to the fruit bowl ➢ Fruits and obstacles will appear randomly. ➢ One point is added for each fruit eaten, and the Meow sound will be played. ➢ One point is deducted each time when an obstacle is encountered. ➢ Win the game when the player gets 5 points.

Design your Scratch Game

How would you like to use Scratch to make a game? Use the flow diagram to design your game.

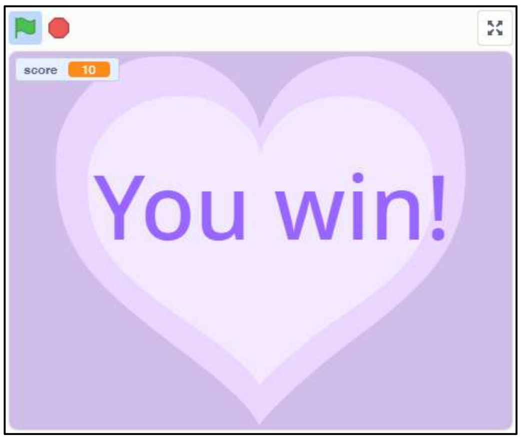
Sample (2) "Fruit Game (Good) for teacher's reference:
https://scratch.mit.edu/projects/859724145/




Instructions

Use the keyboard (up/down/left/right keys) to control the cat to eat fruits.

Win the game when you get 10 points.



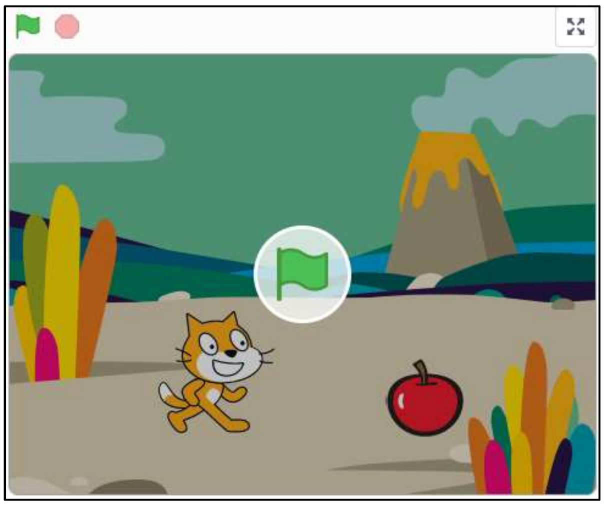
Game Design:

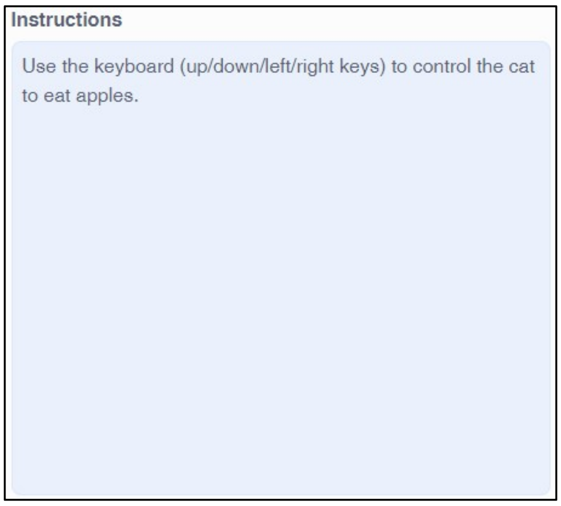
Sprite	Apple, Cat
Motion	up/down/left/right arrow keys, Touch, Score
Design / Rules of the Game	Use the keyboard (up/down/left/right keys) to control the cat to eat fruits. ➢ Fruits will appear randomly. ➢ One point is added for each fruit eaten. ➢ Win the game when the player gets 10 points.

Design your Scratch Game

How would you like to use Scratch to make a game? Use the flow diagram to design your game.

Sample (2) "Fruit Game (Satisfactory) for teacher's reference:
https://scratch.mit.edu/projects/859722460/





Instructions

Use the keyboard (up/down/left/right keys) to control the cat to eat apples.

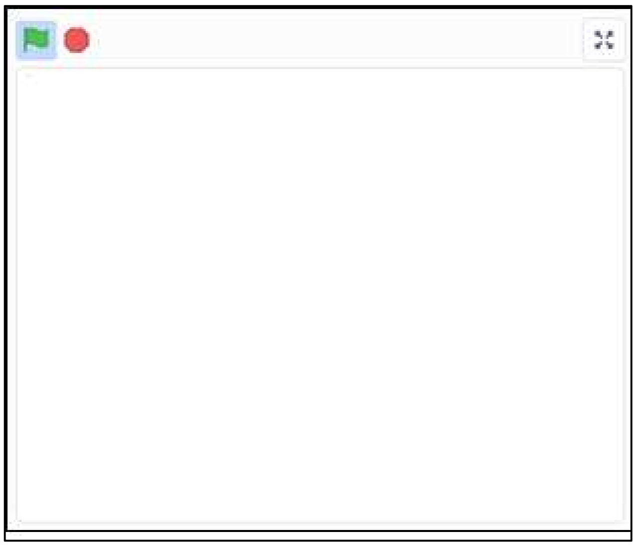
Game Design:

Sprite	Apple, Cat
Motion	up/down/left/right arrow keys, Touch, Score
Design / Rules of the Game	Use the keyboard (up/down/left/right keys) to control the cat to eat apples. ➢ Apples will appear in certain places at any time. ➢ After the cat eats the apple, the apple will disappear.

Design your Scratch Game

How would you like to use Scratch to make a game? Use the flow diagram to design your game.

Game Design:

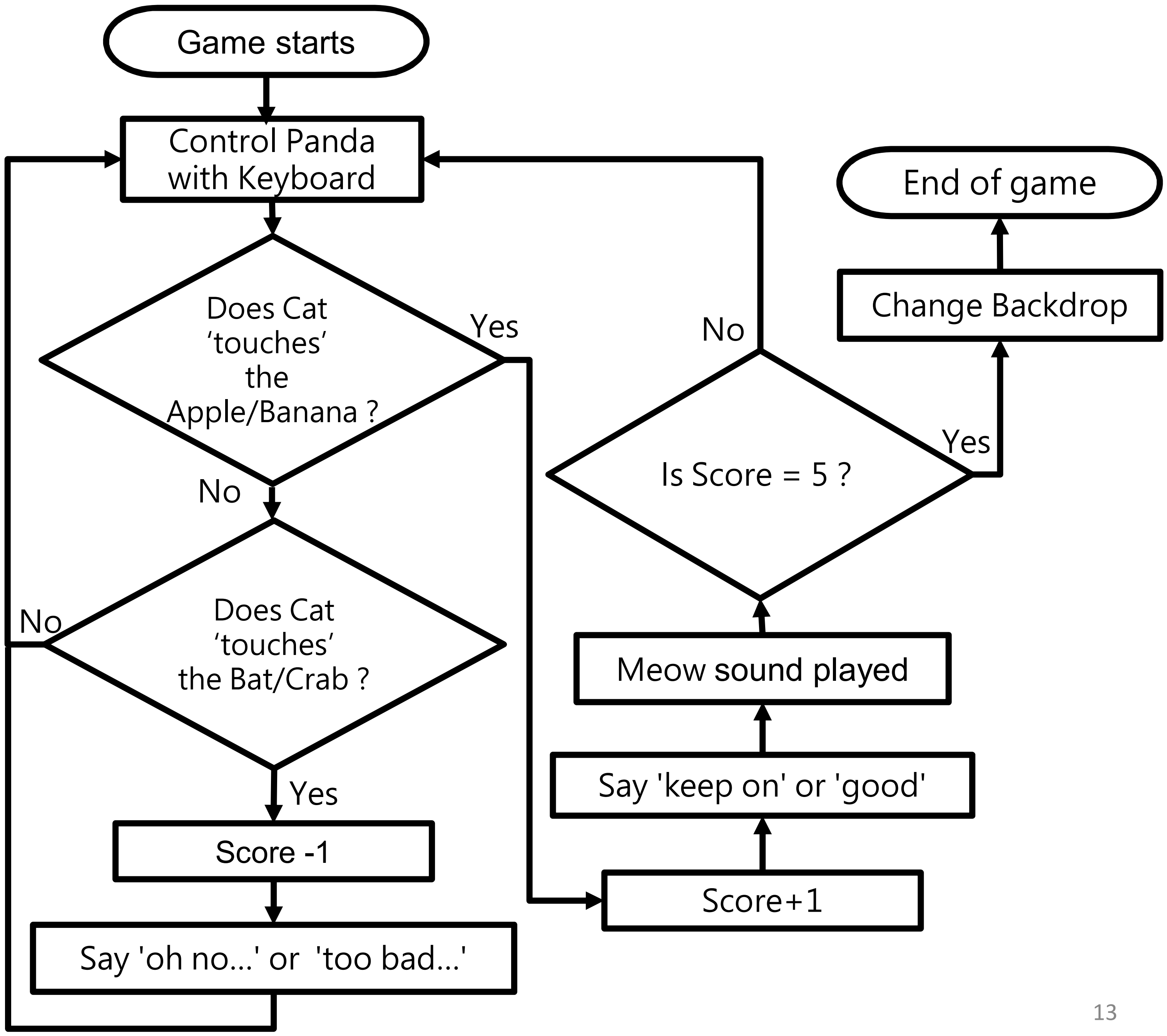


Sprite	
Motion	
Design / Rules of the Game	

Design your Scratch Game

Use the flow diagram to design your game with a starting and ending point. It usually involves conditionals to determine an action.

Flow diagram of Sample (2) “Fruit Game (Outstanding) ” for teacher’s reference.


Game starts
Control Panda with Keyboard
Does Cat ‘touches’ the Apple/Banana ?
Yes
No
Does Cat ‘touches’ the Bat/Crab ?
No
Yes
Score -1
Say 'oh no...' or 'too bad...'
Score+1
Say 'keep on' or 'good'
Meow sound played
Is Score = 5 ?
No
Yes
Change Backdrop
End of game

Design your Scratch Game

Use the flow diagram to design your game with a starting and ending point. It usually involves conditionals to determine an action.

To-Do Checklist

- ❑ Try to make a To-Do Checklist before starting. It is a good practice to keep track of your progress.

  - ❖ Before each lesson, write down what you want to accomplish.
  - ❖ At the end of each lesson, write down what is completed, or what problems prevents you from completing that task.

Date/Lesson	Tasks to be completed	Status (completed / encountered problems, how you will fix it)
e.g. Lesson 1	**Design the program (algorithm) with a flow diagram.**	**Completed on Lesson 1**
Lesson 2	**Add the sprites and backdrop in Scene 1**	**Completed on Lesson 2**

To-Do Checklist

Date/Lesson	Tasks to be completed	Status (completed / encountered problems, how you will fix it)

Pair up with a group

Peer Assessment Worksheet

Pair up with a group. Listen to the other group's design carefully. Try to provide them some constructive feedback for a better design!

For example,

👍 *The* ***backdrop is*** *beautiful.* 👍 *The* ***sprites*** *are vivid and fun.*

💡 *You can change the cat* ***sprite****'s costumes.* 💡 *Please add* ***sound*** *effect.*

Group ( )

Please write down two things that you like about this project.

👍 ______________________________

👍 ______________________________

Other Suggestions

💡 ______________________________

Group ( )

Please write down two things that you like about this project.

👍 ______________________________

👍 ______________________________

Other Suggestions

💡 ______________________________

Group Presentation

❏ Following your teacher's instructions, you may first enhance your project by using the feedback from "Peer Assessment Worksheet". Then present and share your project by using the Scratch Studio.

❏ When you present your project, you should introduce:

1. Yourself and your groupmates:

   ____

2. The theme / goal of the project (the problem you identified and solved):

   ____

   ____

3. What you are most proud of your project:

   ____

4. Difficulties you overcome.

   ____

Group Presentation Peer Assessment Worksheet

Pair up with a group. Listen to the other group's design carefully.
Try to provide them some constructive feedback for a better design!

For example,

👍 *The* ***backdrop*** *is beautiful.* 👍 *The* ***sprites*** *are vivid and fun.*

💡 *You can change the cat* ***sprite****'s costumes.* 💡 *Please add* ***sound*** *effect.*

Group ( )

Please write down two things that you like about this project.

👍 ______________________________

👍 ______________________________

Other Suggestions

💡 ______________________________

Group ( )

Please write down two things that you like about this project.

👍 ______________________________

👍 ______________________________

Other Suggestions

💡 ______________________________

Reflection

Self-assessment

❏ Please tick ☑ and fill in the blanks as appropriate.

1. Did you meet your own goals for your project?

☐ Yes ☐ I can do better

2. If you could redo your project, how would you enhance the project? Would you change the design or codes of the project?

☐ I want to change the design, because ____________________

☐ I want to change the codes, because ____________________

____

3. What was your role as a partner in this project?

☐ Group Leader ☐ Programmer ☐ Graphic Designer ☐ Others ___

Think about what you did?

____

4. Are you a good partner? Did you respect and inspire your partner in the project?

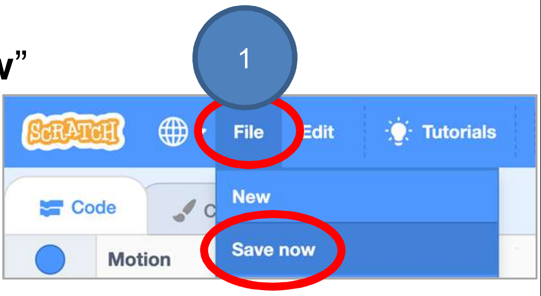
☐ Yes ☐ I can do better

Because I… ____________________

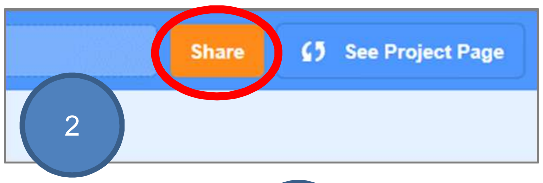
Sharing to Studio for Group Presentation

When you finish, you will add your project to your teacher's Studio.

1. Save your project by clicking "**Save now**" under the File menu.

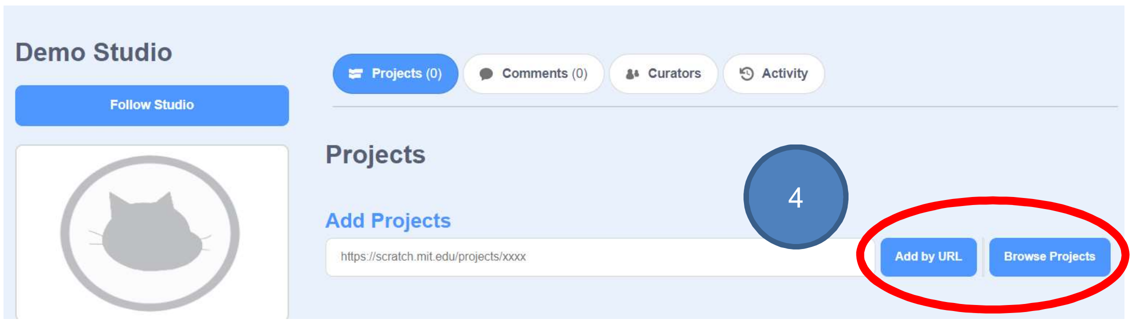


2. Click the orange "**Share**" button.

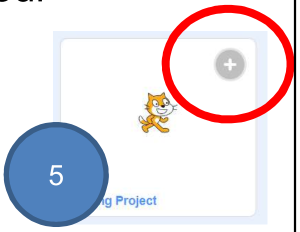


3. Go to your teacher's Studio (they will give you a URL).

4. In the "**Add projects**" column, you can **Add by URL** or **Browse Projects**.



5. If you choose **Browse Project**, then you will see all your shared projects.
   Find the right one and click the "+" to add it to studio.

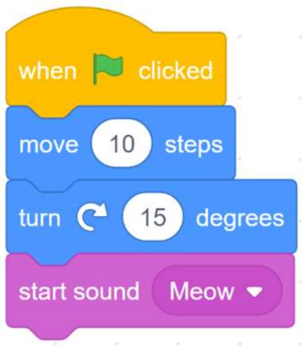


Review Basic Programming Constructs

As you plan your project, think about the basic programming constructs, the key concepts and practices that you have learnt, as well as the blocks you have used before. Try to use various Scratch blocks you've learned so far.

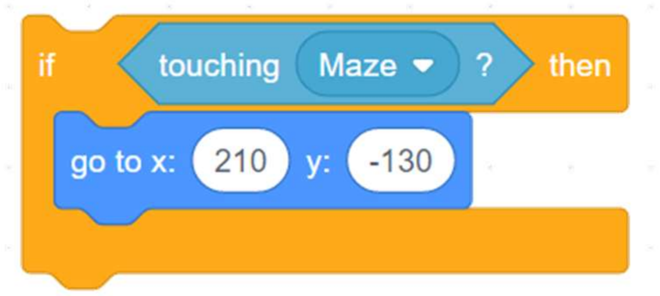
SEQUENCES

Multiple instructions put together form a sequence that happens in order.



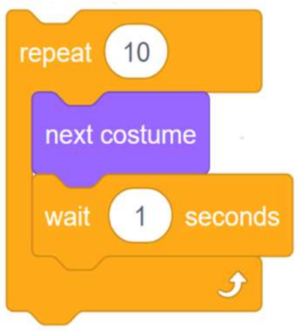
BRANCHING / SELECTION

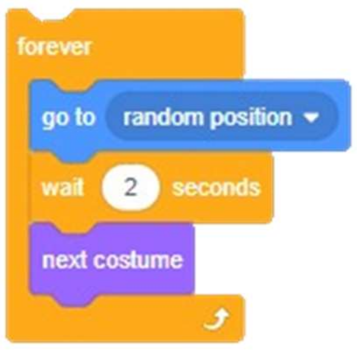
Sometimes you need to make decisions. Trigger sprite action based on if-then blocks.



ITERATION / REPETITION

Repeat and forever blocks let your sprite easily do things over and over.

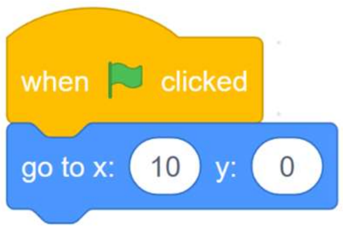




Review Key Concepts and Practices

EVENTS

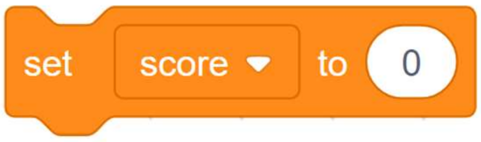
Control sprites based on an event like a key press.



```
when [flag] clicked
go to x: (10) y: (0)
```

VARIABLES

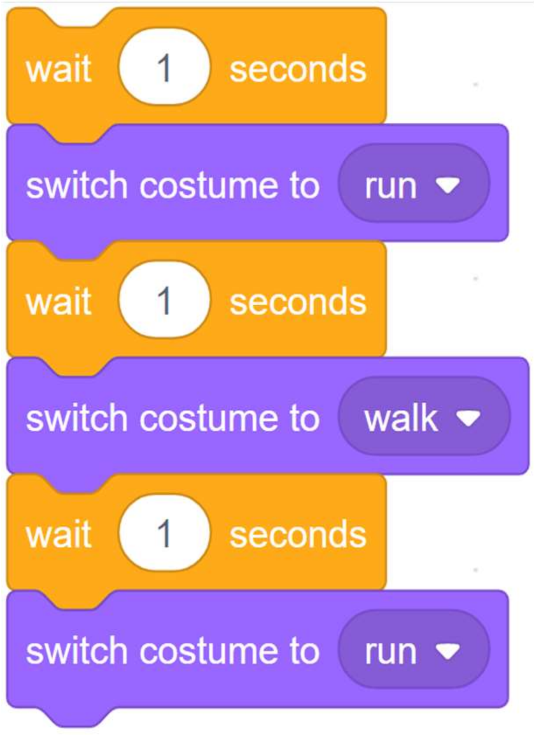
Variables can help you save information and update it in your Scratch project.



```
set [score ▾] to (0)
```

COSTUME CHANGES

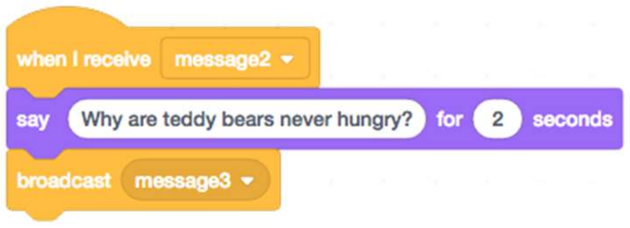
Show movement by changing a sprite's costume!



```
wait (1) seconds
switch costume to [run ▾]
wait (1) seconds
switch costume to [walk ▾]
wait (1) seconds
switch costume to [run ▾]
```

BROADCAST

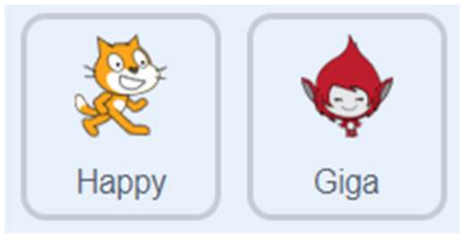
Broadcast and **when I receive** blocks let sprites talk to each other to trigger actions.



```
when I receive [message2 ▾]
say (Why are teddy bears never hungry?) for (2) seconds
broadcast [message3 ▾]
```

NAMING

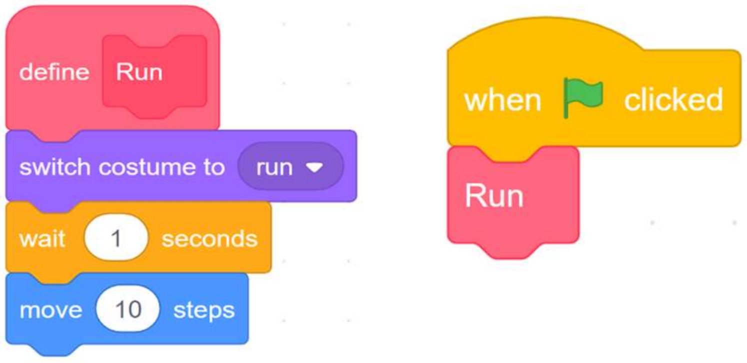
Give an good name to the sprites is also important.



Review Key Concepts and Practices

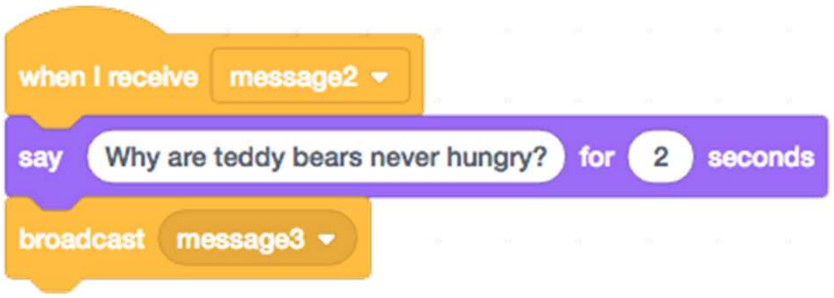
MY BLOCKS

Use custom blocks (My Blocks) to break complex tasks into smaller blocks of code that can be reused.


```
define Run
switch costume to run
wait 1 seconds
move 10 steps

when [flag] clicked
Run
```

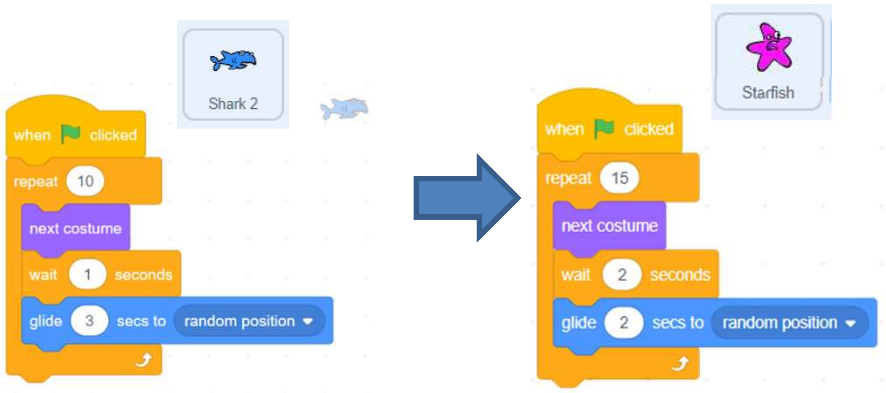
BROADCAST

Broadcast and **when I receive** blocks let sprites talk to each other to trigger actions.


```
when I receive message2
say Why are teddy bears never hungry? for 2 seconds
broadcast message3
```

REUSE AND REMIX PROGRAMS/ CODES

We can reuse and remix the codes of one sprite and use them for the second and third sprites.

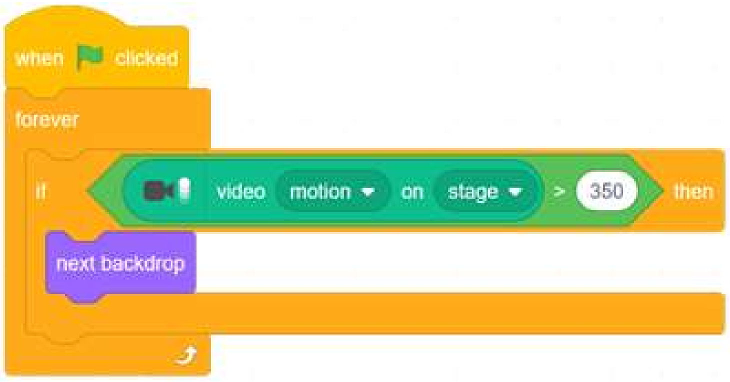


```
when [flag] clicked
repeat 10
    next costume
    wait 1 seconds
    glide 3 secs to random position

when [flag] clicked
repeat 15
    next costume
    wait 2 seconds
    glide 2 secs to random position
```

Review Key Concepts and Practices

CONDITIONAL OPERATORS

We use **operators** to evaluate whether a condition is true or false. Conditional expressions always use operators such as greater than (>), less than (<) or equal (=).

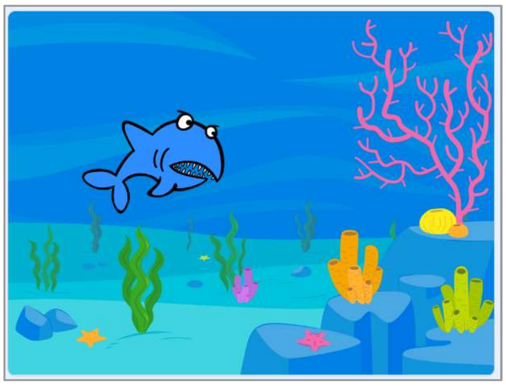
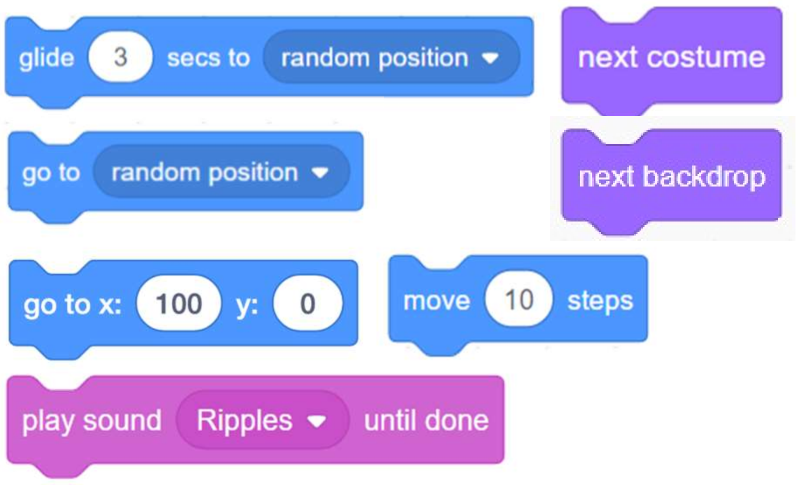
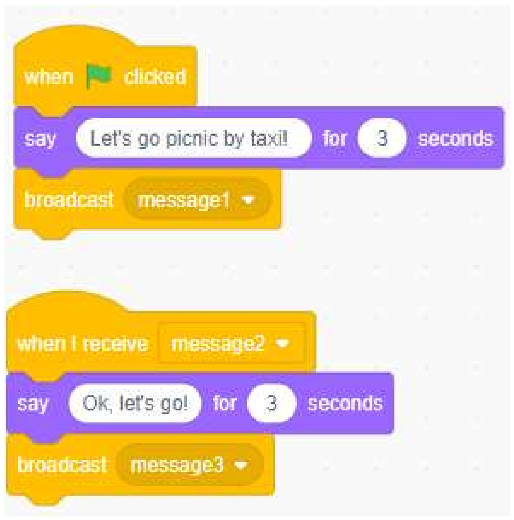
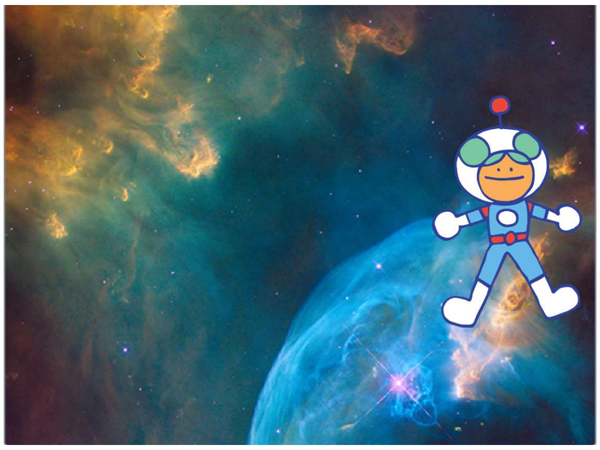
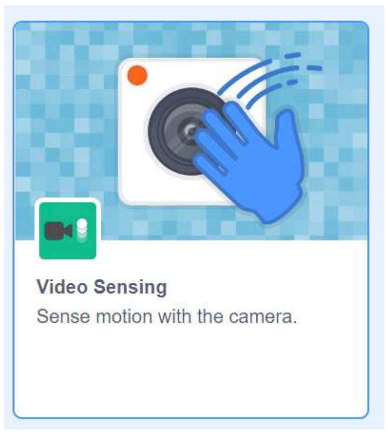
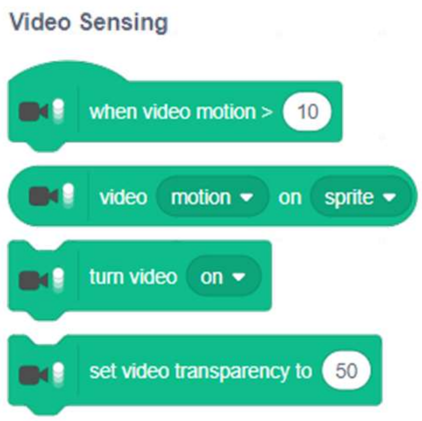


```
when [flag] clicked
forever
    if <(video motion on stage) > 350> then
        next backdrop
```

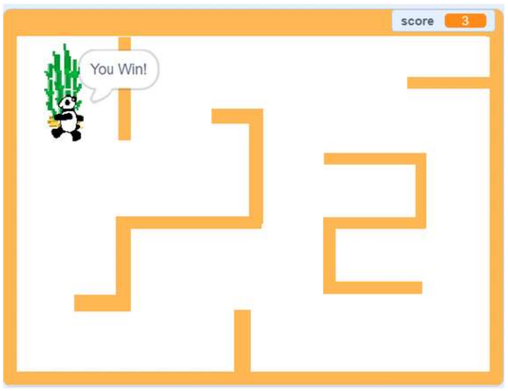
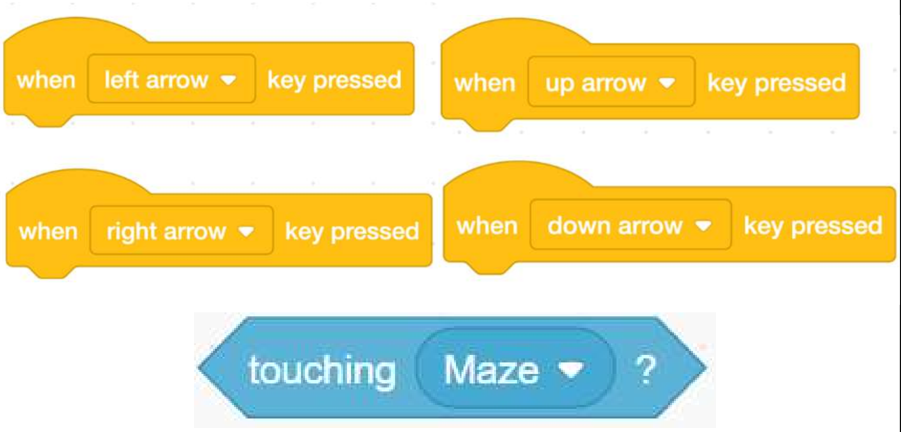
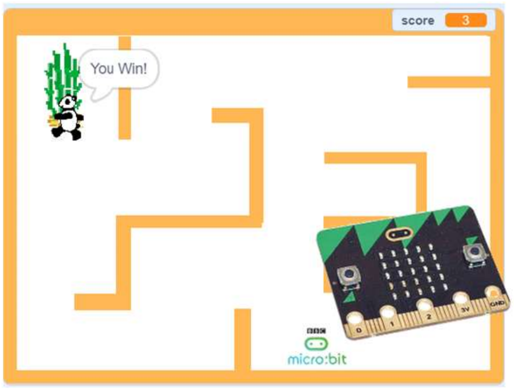
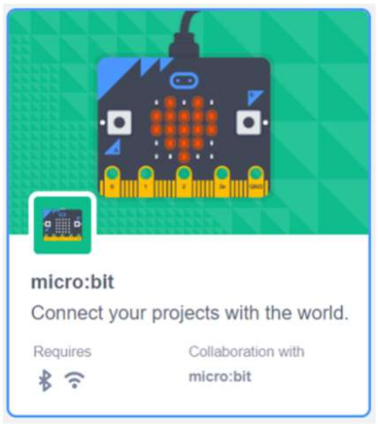
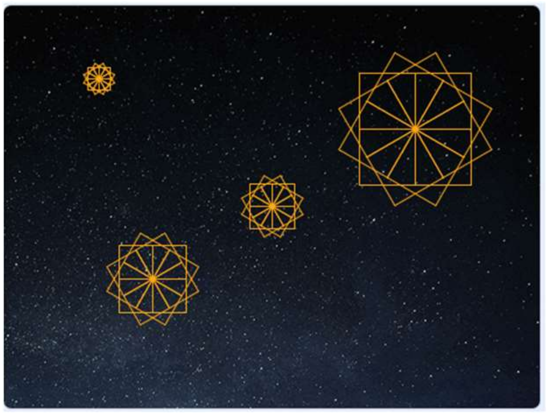
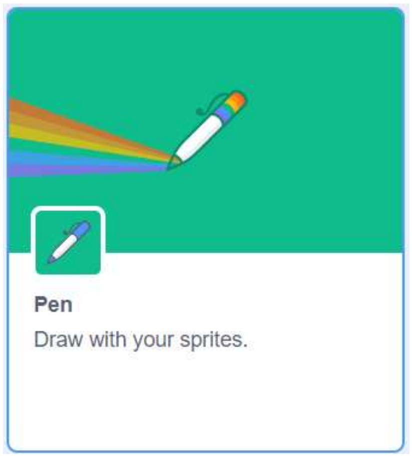
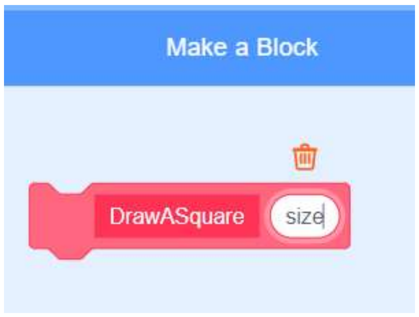
Appendix:

Review Scratch Features

- ❑ Let's review what features you have learnt so far in the previous Scratch units! You can play it again anytime!

Unit	**Key Features Learnt**
Unit 2  https://scratch.mit.edu/projects/722781437	 glide (3) secs to (random position) · next costume · go to (random position) · next backdrop · go to x: (100) y: (0) · move (10) steps · play sound (Ripples) until done
Unit 3 We are all eco-warrior! We are all eco-warrior! We are all eco-warrior! https://scratch.mit.edu/projects/731740461/	 when 🏳 clicked say (Let's go picnic by taxi!) for (3) seconds broadcast (message1) when I receive (message2) say (Ok, let's go!) for (3) seconds broadcast (message3)
Unit 4  https://scratch.mit.edu/projects/727401089	 Video Sensing Sense motion with the camera.  Video Sensing when video motion > (10) video (motion) on (sprite) turn video (on) set video transparency to (50)

Review Scratch Features

Unit	**Key Features Learnt**
Unit 5  https://scratch.mit.edu/projects/722154863	 when left arrow key pressed when up arrow key pressed when right arrow key pressed when down arrow key pressed touching Maze ?
Unit 6  https://scratch.mit.edu/projects/734787236/	 micro:bit Connect your projects with the world. Requires Collaboration with micro:bit
Unit 7 and 8  https://scratch.mit.edu/projects/737985288	 Pen Draw with your sprites.  Make a Block DrawASquare size

Project Assessment Rubric (For Project Completed in Groups)

Category	Outstanding	Good	Satisfactory	Work Harder
Problem Identification for the project (16%)	Able to identify and define the problem clearly for the project through careful observation and concrete analysis; and propose a relevant and feasible solution / design with careful plan to address the specific problem. Score: 13 - 16	Able to identify and define the problem for the project through observation and concrete analysis; and propose a feasible solution / design with proper plan to address the problem. Score: 9 - 12	Generally able to identify and define the problem for the project; and propose a feasible solution / design with a brief plan to address the problem. Score: 5 - 8	The problem is not clearly identified and defined; and the proposed solution / design is not relevant to address the problem. Score: 0 - 4
Algorithm Design with Application of Computational Thinking Concepts and Practices (40%)	The program is in a logical sequence. It is designed in a highly effective way to accomplish the task fully, with outstanding use of iteration, branching / selection, variables, etc., to further simplify and optimize the program. Score: 31 - 40	The program is in a logical sequence. It is effective to accomplish the task fully, with good use of iteration, branching / selection, variables, etc., to simplify and optimize the program. Score: 21 - 30	The program is, in general, in a logical sequence. It can accomplish the task to a satisfactory extent with generally appropriate use of iteration, branching / selection, variables, etc. Score: 11 - 20	The program lacks logical sequence. It lacks the use of iteration, branching / selection, variables, etc. The task cannot be accomplished. Score: 0 - 10
Application of Features and Blocks in the Scratch Programming Platform (20%)	Excellent use of features and blocks of Scratch to create a highly interactive and actively engaging program in the context of identified problem. Score: 16 - 20	Good use of features and blocks of Scratch to create an engaging program in the context of identified problem. Score: 11 - 15	Appropriate use of features and blocks of Scratch to create a program that can work properly in the context of identified problem. Score: 6 - 10	Limited use of features and blocks of Scratch to create a program in the context of identified problem and fail to achieve the expected outcome. Score: 0 - 5
Aesthetics (12%)	The project showcases	The project shows creativity and has a	The project attempts to show creativity	The project does not show creativity or

	outstanding creativity with high level of visual and sensory appeal. Good attention is given to address the details. It fully engages the senses of the users and successfully enhances users' experience. Score: 10 - 12	good visual design. There is good use of sound, visual sensory elements to engage the users. Good attention is given to address the details. Score: 5 - 9	and has appropriate visual design. There is appropriate use of sound, visual sensory elements to engage the users. Score: 4 - 6	visual design. Limited use of sound, visual sensory elements to engage the users. Score: 0 - 3
Collaboration with Peers (12%)	Outstanding communication and collaboration between team members with a strong sense of team spirit. Team members cooperate well to achieve the task effectively. Every team member has a clear role to play and works together harmoniously towards a shared goal, with a high level of trust and respect for each other. Score: 10 - 12	Good communication and collaboration between team members with a good sense of team spirit. Team members cooperate well to achieve the task. Every team member has a role to play and works together towards a shared goal with trust and respect for each other. Score: 7 - 9	Appropriate communication and collaboration between team members with a sense of team spirit. Every member has a role to play and most team members can cooperate and work together towards a shared goal to achieve the task. Score: 4 - 6	Limited communication and collaboration between team members with scarce sense of team spirit. Most team members work independently and fail to help each other solve problems, resulting in the task not being completed. Score: 0 – 3
Overall Score (Total: 100%) • Outstanding: 76 – 100 • Good: 51 – 75 • Satisfactory: 26 – 50 • Work Harder: 0 – 25				

Project Assessment Rubric (For Project Completed by Individual)

Category	Outstanding	Good	Satisfactory	Work Harder
Problem Identification for the project (20%)	Able to identify and define the problem clearly for the project through observation and concrete analysis; and propose a relevant and feasible solution / design with careful plan to address the specific problem. Score: 16 - 20	Able to identify and define the problem for the project through observation and analysis; and propose a feasible solution / design with proper plan to address the problem. Score: 11 - 15	Generally able to identify and define the problem for the project; and propose a feasible solution / design with a brief plan to address the problem. Score: 6 - 10	The problem is not clearly identified and defined; and the proposed solution / design is not relevant to address the problem. Score: 0 - 5
Algorithm Design with Application of Computational Thinking Concepts and Practices (40%)	The program is in a logical sequence. It is designed in a highly effective way to accomplish the task fully, with outstanding use of iteration, branching / selection, variables, etc., to further simplify and optimize the program. Score: 31 - 40	The program is in a logical sequence. It is effective to accomplish the task fully, with good use of iteration, branching / selection, variables, etc., to simplify and optimize the program. Score: 21 - 30	The program is, in general, in a logical sequence. It can accomplish the task to a satisfactory extent with generally appropriate use of iteration, branching / selection, variables, etc. Score: 11 - 20	The program lacks logical sequence. It lacks the use of iteration, branching / selection, variables, etc. The task cannot be accomplished. Score: 0 - 10
Application of Features and Blocks in the Scratch Programming Platform (20%)	Excellent use of features and blocks of Scratch to create a highly interactive and actively engaging program in the context of identified problem. Score: 16 - 20	Good use of features and blocks of Scratch to create an engaging program in the context of identified problem. Score: 11 - 15	Appropriate use of features and blocks of Scratch to create a program that can work properly in the context of identified problem. Score: 6 - 10	Limited use of features and blocks of Scratch to create a program in the context of identified problem and fail to achieve the expected outcome. Score: 0 - 5
Aesthetics (20%)	The project showcases	The project shows creativity and has a	The project attempts to show creativity	The project does not show creativity or

	outstanding creativity with high level of visual and sensory appeal. Good attention is given to address the details. It fully engages the senses of the users and successfully enhances users' experience. Score: 16 - 20	good visual design. There is good use of sound, visual sensory elements to engage the users. Good attention is given to address the details. Score: 11 - 15	and has appropriate visual design. There is appropriate use of sound, visual sensory elements to engage the users. Score: 6 - 10	visual design. Limited use of sound, visual sensory elements to engage the users. Inadequate attention is given to address the details. Score: 0 - 5

Overall Score (Total: 100%)

- Outstanding: 76 – 100
- Good: 51 – 75
- Satisfactory: 26 – 50
- Work Harder: 0 – 25