Basic Coding Courses

Lesson 1 The secret of Codey Rocky

Lesson Plan

Overview:

The goal of this lesson is to help students understand the concept of **Program** and what programs can do, as well as the basics of Codey Rocky and mBlock 5.

Teaching Objectives:

1. Understand the definition of Program and what Program can do.
2. Get to know Codey Rocky and its features.
3. Master the basics of mBlock 5.
4. Learn how to upload programs.

Preparation:

1. A whiteboard and a whiteboard marker (or you can use a blackboard and chalks)
2. One Codey and a Bluetooth dongle (or the USB cable) per student but it's fine if 2 or 3 students share one set.
3. A computer with installed mBlock 5 for each student but it's fine if 2 or 3 students share a computer.

Prior Knowledge

1. Computer basics;
2. Basic cognitive skills.

Teaching Procedure:

1. Warm up —— The Secret of Codey Rocky

Introduce to students what Codey Rocky is: it's a tiny yet versatile robot.

The teacher can demonstrate the features of Codey Rocky through videos. Or the teacher can upload the programs to Codey Rocky in advance, making the robot perform such tasks as avoiding obstacles, following lines and more.

Ask students: **Apart from Codey Rocky, can you think of any other robots? What are those robots used for?** Pick one of the students to answer the question.

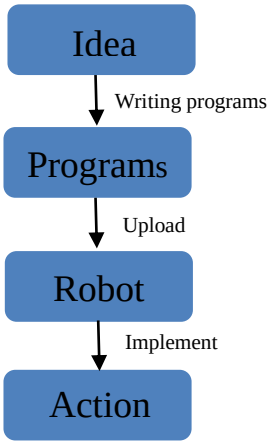
Sample answers: delivery robots, robots in logistics, security robots…

Ask again: How can those robots understand our instructions?

Give students some time for discussion.

2. New Concept —— Program

The teacher says: Program is an artificial language that we use to tell robots what to do. We translate our instructions into a piece of program. Then we upload the program to the robot, making it do a variety of things as programmed.



Ask: Do you know what the answer is now? What's the secret of Codey Rocky?

Possible answer:

It's because Codey Rocky is uploaded with programs written by us.

3. Demonstration

The teacher asks students: Do you want to write code and upload the code to Codey Rocky? But before that, you need to get to know Codey Rocky. It's an adorable yet powerful robot.

Codey Rocky is an educational programmable robot. You can use software to code the robot, manipulating it to do a variety of things you can imagine. It's also a good companion that can help children learn to code. With mBlock 5, children are able to master the basics of coding and develop logical thinking as well as computational thinking. Also, Codey Rocky supports technologies like AI and IoT, which exposes children to the latest cutting-edge technologies.

Tell students: Combine Codey with Rocky and then you get a Codey Rocky. Now let's take a look at them one by one.

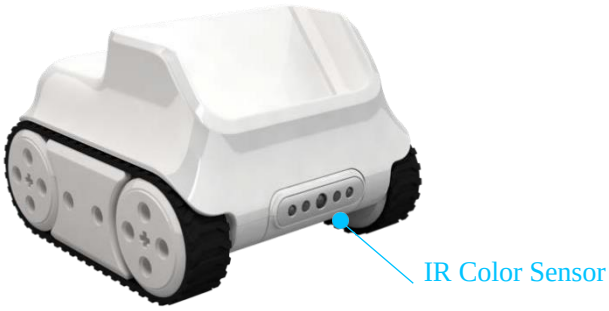
1) Codey: As the brain of the robot, Codey is equipped with a variety of sensors and programmable blocks. It can work individually and can also work with Rocky to perform more tasks. Now pick up your Codey. Let's take a look at what sensors it has.

Tips:

You can use the picture above to show students what sensors Codey Rocky has and what purposes those sensors are used for.

Feature	Block Name	Applications and Core Functions
Input	IR transmitter and IR receiver	Facilitates communication between Codeys and remote controls
	Gear potentiometer	Adjusts the input signals
Input	Gyroscope	Detects how Codey moves and the angles.
	Buttons	Buttons can be programmed to control the facial expressions, motions, and sounds of Codey Rocky.
	Light Sensor	The sensor is used to measure the volume of sounds in the surrounding environments.
	Sound Sensor	The sensor is used to measure the light intensity of surrounding environments.
Output	Speaker	The sensor can be programmed to play music.
	RGB indicator	The indicator can glow in different colors.
	LED matrix screen	The images, texts and time displayed on the screen are all customizable.

2) Rocky serves as the chassis of Codey. It adds more abilities to Codey, like avoiding obstacles, identifying colors, following lines and more.



Tip:

Show the picture above to students. Let students know what sensors Codey Rocky has and what purposes those sensors are used for.

Features	Name	Applications
Input	IR Color Sensor	The IR Color Sensor integrates a color sensor, a grayscale sensor, an IR proximity sensor. By toggling the IR Color Sensor, you can make Codey Rocky perform a variety of fun tasks, like avoiding obstacles, following lines and more.
Input	Motor	The motor is used to control the motion of Rocky.

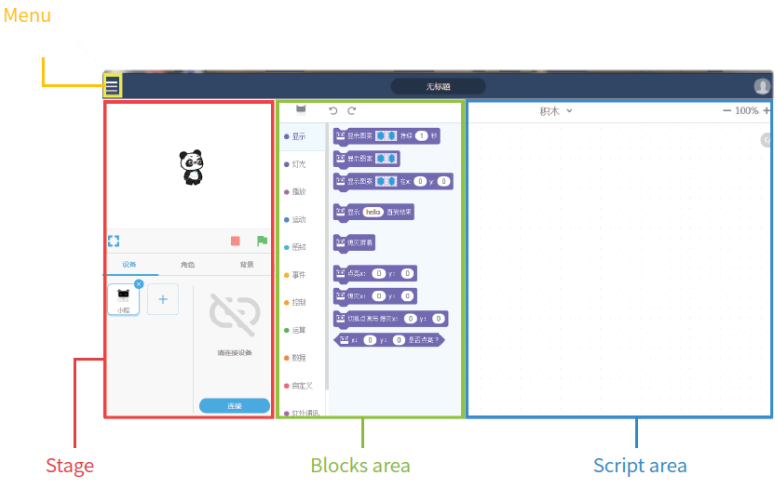
4. Imitate or Create

Have students practice how to code Codey Rocky to move!

Task 1: About mBlock 5

mBlock 5 is a programming tool which supports block-based and Python programming languages. It's developed based on the Scratch 3.0, an open-source software tool that's developed jointly by MIT and Google. Using mBlock 5, you can write programs that tell Codey Rocky or other robots to do whatever you would like. You can even take advantage of the software to create stories, games, and animations that are engaging and unique. Moreover, mBlock 5 exposes children to technologies, like AI, deep learning and model training. In a word, mBlock 5 can be one of the best options for first-time coders.

Have students open mBlock 5 PC and walk them through the interface.



Startup Interface

1. **Stage**: In this area, you can show your projects, connect devices and upload programs, add sprites and backgrounds.

2. **Blocks area**: You can find the blocks you need by color or category.

3. **Script area**: You drag blocks to this area to form programs.

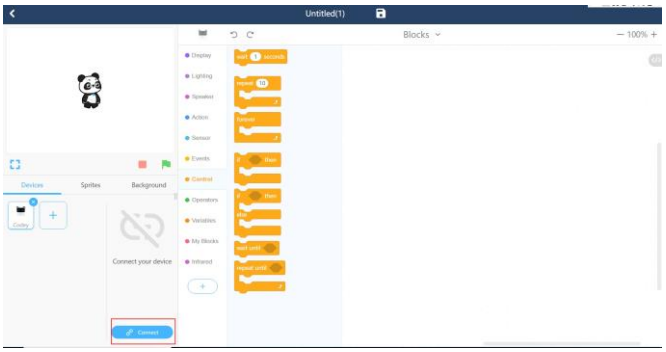
4. **Device/Sprites/Backgrounds Setting area**: From here, you can find the devices, sprites, and backgrounds you need.

Task 2: Bring Codey Rocky to life

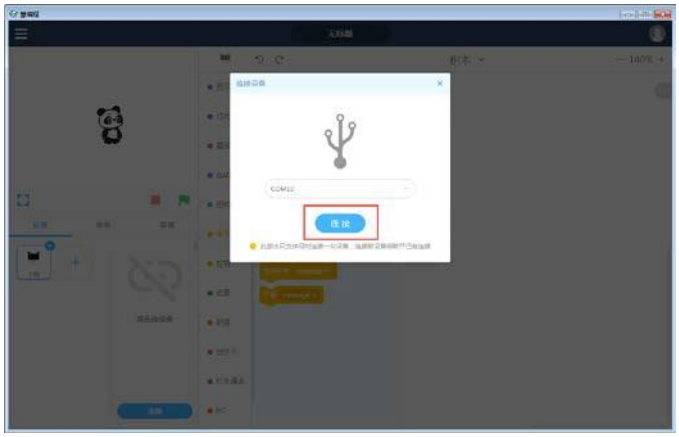
Have students practice how to make Codey Rocky move as programmed.

1. Connecting to a computer: Connect Codey to the computer via the USB cable. Then power on Codey.

2. Selecting the serial port: Open mBlock 5, click Connect, and select the correct serial port.

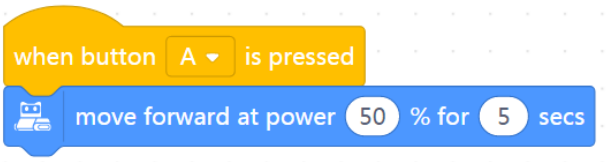


Click Connect



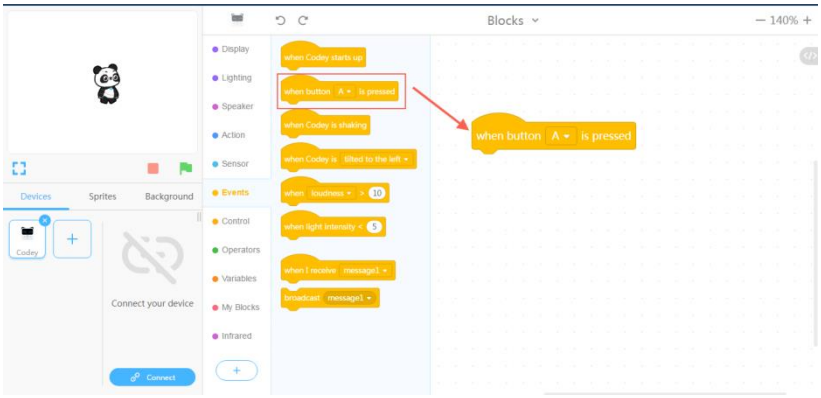
Select the serial port

3. Use mBlock 5 to create a piece of program as shown below:

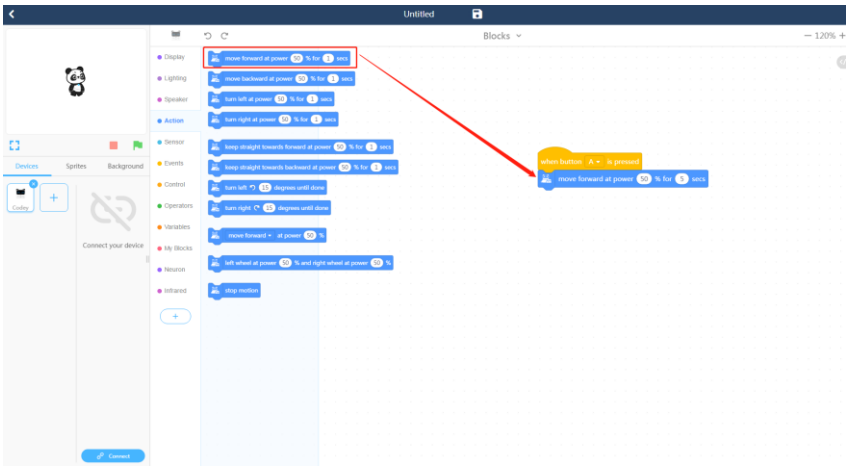


How to do:

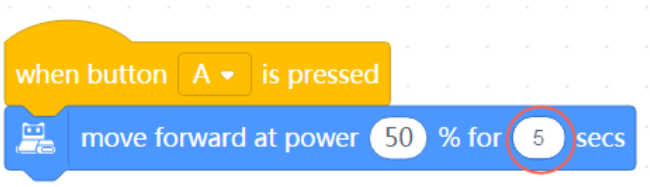
Step 1: Drag the **when button A is pressed** block out of the Event category.



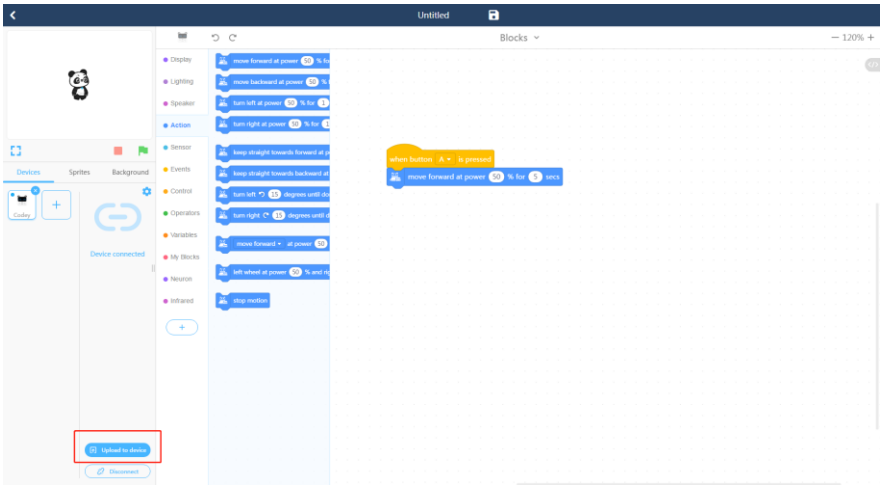
Step 2: Drag the **move forward at power 50% for (1) secs** block out of the **Action** category.



Step 3: Change the time to 5 seconds.



Step 4: Upload the program to Codey.



Click Upload to Device

Step 5. Unplug the USB cable and put Codey Rocky on the table. Press the button A and observe how Codey Rocky reacts. Have students work in pairs to complete the tasks above by writing programs.

5. Presentation

You can show some well-done projects or have volunteers present their works. When sharing the projects, students are supposed to give their answers to the questions proposed by you.

After one student presents his or her project, you can invite some students to comment on the project (what is good about the project and where to improve). Then give your own comments.

6. Wrap up

You need to give a summary of today's lesson.

Program: Program is an artificial language that we use to tell robots what to do. We translate our instructions into a piece of program. Then we upload the program to the robot, making it do a variety of things as programmed.

Ask students: Can you walk me through the interface of mBlock 5?

Sample answer: Stage area, Blocks area, Scripts area, Device/Sprites/Backgrounds Setting area.

7. Students' Self-review

Please find the attached self-review worksheets. Hand out the copies and ask students to spend a few minutes filling the worksheets.

Lesson 2 Press Buttons to Change Emotions -Events

Lesson Plan

Overview:

Learn how to write code using the Event blocks and create different facial expressions.

Teaching Objectives

1. Understand the concept of **Events**.

2. Master how to use the **Events** blocks in a program.

3. Use the **Events** blocks to create buttons that can function as you would like.

Preparation:

1. A whiteboard and a whiteboard marker(or you can use a blackboard and chalks)；

2.One Codey and a Bluetooth dongle (or the USB cable) per student but it's fine if 2 or 3 students share one set;

3. A computer with installed mBlock 5 per student but it's fine if 2 or 3 students share a computer.

Prior Knowledge:

1. Got to know Codey Rocky;

2. Got to know the interface of mBlock 5;

3. Got to know how to write programs by dragging and dropping blocks.

Teaching procedure:

1. Review:

Ask students:

1) What can you learn from Codey Rocky?

2) By learning how to code, what technologies can you master?

Sample answers:

1) Able to master coding;

2) Able to master technologies like AI, IoT and more.

2. Explain New Concepts:

Expose students to the concept of **Event**. Tell students what an Event refers to. For example: When it gets dark and we enter into a room, we need to turn on the light. To turn on the light, we need to press the light button. In this case, pressing the button is an **event** and that the light is turned on is the result.

Invite students to play a game, helping them have a better understanding of the concept.

3. Game——Follow the Instructions

Game rules:

You should:

1) Divide the students into 3 or 4 groups.

2) Draw some figures on the blackboard, like triangle, circle, square and star.

3) Define 3 or 4 events:

1. When you put your hand on the triangle;
2. When you put your hand on the circle;
3. When you put your hand on the square;
4. When you put your hand on the star.

The four events above trigger the actions below:

1. When you put your hand on the triangle —— the 2nd group of students stands up;
2. When you put your hand on the circle—— the 4th group of students stands up;
3. When you put your hand on the square—— the 1st group of students stands up;
4. When you put your hand on the star—— the 3rd group of students stands up.

Game procedure and teaching preparation:

1) Draw figures on the blackboard.
2) Divide students into groups and have them get ready for the game.
3) Put your hands on a shape randomly and check whether students react as required.
4) If students react as you expect, then you put your hand on another shape. If students fail to react as required, you need to repeat the game rules to students.
5) Repeat the game several times and speed up the process of switching between shapes.

You need to give a summary: In this case, the hand serves as an event. When the hand points to one shape, one specific group of students is expected to stand up as required.

Tips:

1) You can define the sequence number of groups;
2) You can customize the event. For instance, you can define the event as pointing to eyes or nose, or clapping hands 2 times or 3 times.

4. Demonstration

You need to demonstrate and give explanations. Then have students practice on their own.

Use the Event block to make Codey【start up and smile】.

Programs:

Explain to students:

The yellow block【when Codey starts up】is the Event: it means when Codey starts up;

The blue block【show image】is used to change the image displayed on the LED panel.

5. Practice

Task 1: Learn how to set events. Write programs to make Codey Rocky change its facial expressions based on the events. (when button A/B/ C pressed).

Tips:

1) Jump to the Task 2 if students are bright.

Have students work in pairs and practice themselves. Assist students when they need help.

Task 2: Write programs to make Codey Rocky react in response to the events (when button A/B/C pressed), like changing its facial expression or making different sounds. Have students share their projects.

Tip：

1) If time allows, you can have students edit the images as they would like.

You can tailor the time limit based on your teaching purposes and the characteristics of students. Recommended time: 10 min

Here are the sample projects:

Program Story:

【When button A is pressed】, then【play sound switch】and Codey open its eyes;

【When button B is pressed】, then 【play sound switch】and Codey smiles；

【When button C is pressed】, then【play sound switch】and Codey becomes sad.

Have students complete the tasks above in the form of pair programming.

Tell students that there will be a presentation session. Students are expected to share their projects with the whole class and give their answers to the following questions:

1) What is your project about?

2) Did you come across any problems? How did you solve them?

6. Presentation

You can show some well-done projects or have volunteers present their works. When sharing the projects, students are supposed to give their answers to the questions proposed by you.

After one student presents his or her project, you can invite some students to comment on the project (what is good about the project and where to improve). Then give your own comments.

7. Wrap up

You need to give a summary of today's lesson:

Remind students that an event is the beginning of a piece of program. When you write programs, the first thing you need to do is to select an event.

8. Students' Self-review

Please find the attached self-review worksheets. Hand out the copies and ask students to spend a few minutes filling the worksheets.

Lesson 3 To be an Animation Designer

Lesson Plan

Overview:

Understand the concept of Sequence and write programs to create animations.

Objectives:

1. Understand the concept of **Sequence**.

2. Master the basics of Sequence and learn how to create animations using **Sequence**.

Preparation:

1. A whiteboard and a whiteboard marker(or you can use a blackboard and chalks)；

2.One Codey and a Bluetooth dongle (or the USB cable) per student but it's fine if 2 or 3 students share one set;

3. A computer with installed mBlock 5 per student but it's fine if 2 or 3 students share a computer;

4. Copies of A4 paper and pens (the amount depends on the number of students).

Prior Knowledge:

1. Mastered the basics of coding;

2. Knew how to use mBlock 5.

Teaching Procedure:

1. Review last lesson.

Ask students:

1) What is an Event?

2) Can you think of any events in daily life?

3) What events were used in the last lesson?

Sample answers:

1) Event is an action that can cause things to happen;

2) Pressing the button leads to the light bulb lighting up.

In this case, pressing the button is an event and that the light bulb is turned on is the result.

3) Events that are used in the last lesson include: when program starts up, when button A/B/C pressed.

2. Explain New Concept—— Sequence

1) Ask students: What are the steps to put a watermelon into a refrigerator?

2) Sample answer: Open the refrigerator door and put the watermelon into the refrigerator. Students might give you other answers.

3) Explain to students: To put the watermelon into the refrigerator, you need to take these steps: open the refrigerator, put the watermelon into the fridge, close the fridge door. If you are not taking the steps, you will be not able to put the water into the fridge.

4) Ask students: Can you think of any cases in which you need to follow a set of steps to achieve something?

Tip:

Each example should only offer one specific order. It means that only when you follow one

Leave some time for students to brainstorm. You can give an example: To drink water, you need to uncap the bottle, pour the water into your mouth and screw the cap of the bottle. If you don't take the steps, you won't be able to drink the water.

Have students play a game, helping them have a better understanding of the concept of Sequence.

3. Game ——I'm a Robot

Game rules:

You should act as a robot, walk from somewhere in the classroom to the blackboard and draw a smiley face on it. Invite students to give instructions (move forward, turn left, turn right and more) to the robot and write instructions on paper. You need to follow the instructions.

Game procedure and teaching preparation:

1. Introduce game rules to students and get students ready for the game.

2. Invite students to give instructions and write down the instructions on the A4 paper. Ask students to ensure instructions are arranged in the correct order (from top to bottom).

3. You should read the students' instructions from top to bottom and perform tasks as instructed.

Tips:

1. If students write instructions from left to right, teachers should still read instructions from top to bottom. In this case, there is a possibility that instructions can only be read from left;

2. When students' instructions are unclear, you still need to follow the instructions to make actions. For example, if the instruction is: turn left, move ahead by 4 meters, then the robot should execute the instruction like: turn left and move ahead. This is exactly how the software instructions are performed. When there is no specific setting for the time and the angle, the computer will read the simple instruction for turning left promptly and then read the instruction for moving ahead;

3. If it's necessary to make the instructions more specific, you can remind students about the fact that the robot lays down its two hands vertically. Therefore, when students are giving certain instructions, they need to make sure that the instructions are detailed enough. For instance, if the instruction for the robot is to pick up a pen, the instruction must include details: by which hand, the hand gesture, where to draw the smiley face exactly on the blackboard, etc.;

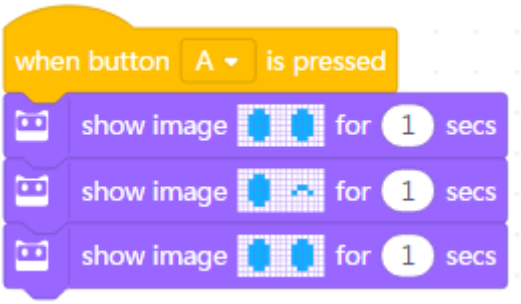
4. In consideration of the time limit and the ages of students, you can simplify the instructions. Anyway, the key point is clear: you need to make instructions specific and arrange them in the correct order if you want the robot to do things as you program.

Summary: When we are programming, we arrange the blocks in the order from top to bottom to form a set of steps. In this way, robots can follow the steps to perform a task. We refer to the set of steps as **Sequence.**

4. Demonstration

Create animations using the block 【show image ( ) for ( ) secs】. The method is simple: Use the image of the block as the base, duplicate the block, and change the image slightly. Repeat the steps and then arrange those blocks in sequence.

Sample project: Winking Eyes



Program story:

Drag out the block【show image ( ) for ( ) secs】and edit the image to be a pair of open eyes;

Duplicate the block【show image ( ) for ( ) secs】but make the eyes wink;

Duplicate the block 【show image ( ) for ( ) secs】once again, and this time make the winking eye open.

Add the Event block ——"When button A pressed"

Upload the programs to the device. When the button A is pressed, Codey will wink at you.

Explain to students:

Traditionally, animation designers would make animations by following these steps: Put a sheet of static drawing on the table first and unfold a new drawing paper on top of the first paper. Designers would outline the frame and then change the drawing bit by bit at a time. Then another piece of paper, outline the frame and change the drawing slightly again. Designers repeat the steps over and over again until they complete a series of pictures that are slightly different from each other. Then, they flip the drawings quickly to animate the pictures. Based on the same principle, we use the block "show image ( ) for ( ) secs" in our program to create animations.

5. Practice

Have students complete the following tasks.

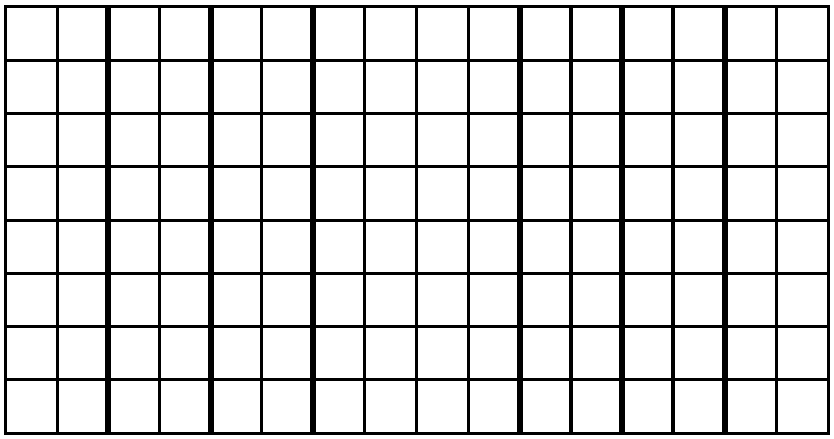
Task 1: Students are supposed to complete the project 【Winking Eyes】as the teacher did.

Have students work in pairs and complete the task as you just did.

Task 2: Create Animations

Students are supposed to share their projects after they complete this task.

Ask students to work in pairs to complete Task 2. Students need to create an image by filling the following grid. Use the block【show image ( ) for ( ) secs】to create the same image; duplicate the block but change the image slightly; repeat the steps until a series of coding blocks are created. These coding blocks show different images and should be arranged in sequence.



Upload the program to Codey. Then play the animation and have students share their projects with classmates.

Tips:

1. Some students might accomplish the task ahead of time. Tell them to give their animations different events;

2. Or have them change how long the animation lasts;

3. If some students fail to accomplish the task on time, invite them to share one thing that happened when they were writing programs. It could be something funny, a challenge they confronted or a problem they have;

4. Students can share their works with the whole class, or they can take turns presenting the works to other groups ;

5. Teachers can tailor the time limit according to teaching purposes and the personality of the class. Recommended time: 10 min

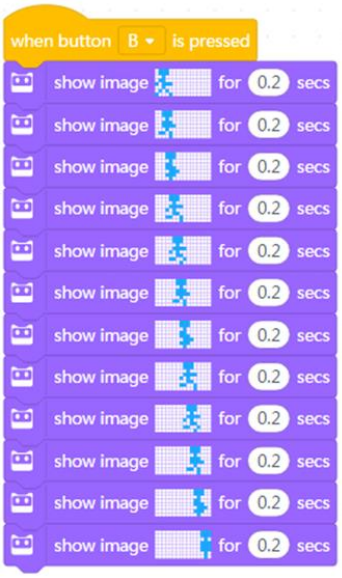
Before students practice on their own, you can show them two sample projects: A *Growing Tree* and *Taking a Walk*.

Sample project: A Growing Tree

Program Story:

Stand up the LED matrix display on the table. When the button B is pressed, the sapling will be growing.

Sample Project: Taking a Walk



Program Story：

When the button B is pressed, a kid will keep running from one side of the LED matrix display all the way to the other side.

Have students complete the tasks above in the form of pair programming.

Tell students that you will invite groups to present their works. And they need to answer these questions:

1) What is your project about?

2) Did you come across any problems? How did you solve them?

6. Presentation

You can show some well-done projects or have volunteers present their works. When sharing the projects, students are supposed to give their answers to the questions proposed by you.

After one student presents his or her project, you can invite some students to comment on the project (what is good about the project and where to improve). Then give your own comments.

7. Wrap up

You need to give a summary of today's lesson:

Remind students that Sequence refers to a series of steps which are carried out in order to complete a task. Sequence.

8. Students' Self-review

Please find the attached self-review worksheets. Hand out the copies and ask students to spend a few minutes filling the worksheets.

Lesson 4 Identify the Bug

Lesson Plan

Overview:

Understand the concept of **Bug** and learn how to identify bugs and fix the bugs.

Teaching objectives:

1. Understand the concept of Bug;

2. Know how to identify bugs and fix the bugs.

Preparation:

1. A whiteboard and a whiteboard marker(or you can use a blackboard and chalks)；

2. One Codey and a Bluetooth dongle (or the USB cable) per student but it's fine if 2 or 3 students share one set;

3. A computer with installed mBlock 5 per student but it's fine if 2 or 3 students share a computer;

Prior Knowledge:

1. Mastered the basics of coding;

2. Mastered the basics of mBlock 5;

3. Knew how to write programs.

Teaching procedure:

1. Review

Ask students:

1) What is Sequence?

2) Can you think of sequences in daily life?

Sample answers:

1) Sequence refers to a series of steps which are carried out in order to complete a task. For example: putting the watermelon into the refrigerator, washing hair and more.

2. Explain the new concept

Tell students what bug refers to.

Bugs are inevitable when we write programs. A "bug" means an insect in its literal sense. But in programming, a bug refers to a mistake that will lead to the failure of a program. It's like a typo in an article. You need to correct the errors because they might make your article difficult to understand.

3. **The Behind Story**

Tell a story to help students have a clearer understanding of the concept of **Bug**.

At the initial stage where computers were invented, unlike today's portable computers, they were extremely large in size back then. (Picture from Wiki).

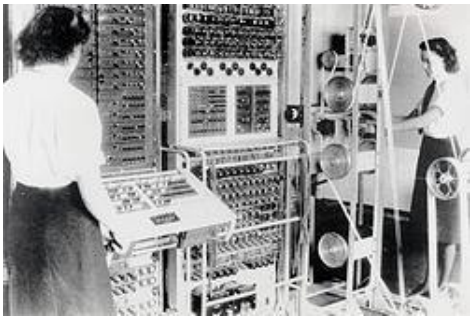


Figure 1 Colossus computers were used to decipher German codes during WW II

Once upon a time, a colossus computer failed to work. The whole team of programmers tried to identify the problem but to no avail. In the end, Grace Murray Hopper, a female programmer, identified what the problem was: a moth flew to the inside of the computer and caused the glitch. When they removed the moth, everything was back on the right track. It was the first bug in a computer program that was found and programmers affixed it to the logbook (see the picture above). Since then, the term bug becomes common in use when people refer to mistakes in a computer program. Naturally, Grace Murray Hopper was since then considered as the Mother of Debug.

Figure 2 The moth caused the computer mistake and this is the first mistake in a computer program.

Move on to the next session. Have students learn how to identify bugs in programs.

4. Demonstration

Show students 3 sample projects:

Sample project 1 - The Car Key was Stolen!

Sample project 2 - The Bomb Can't Count Down!!

Sample project 3 - The Earthworm and the Bug

Remind students of one thing: In order to identify bugs in the programs, we need go over each line of code from top to bottom.

Task 1: Find bugs in the following programs and try to fix them:

1. The Car Key was Stolen!

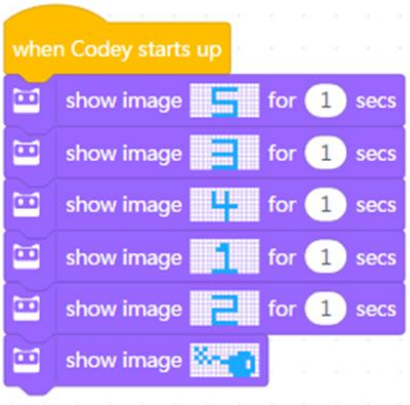
The teacher: tell students that the car key was stolen so they can't start the car. (lack of an event)

Students: Examine the program and try to find the bug.

The teacher: Ask students whether they identify the bug or not.

Students: The bug is found. Students need to add an Event block to the code.

2. The Bomb Can't Count Down!!



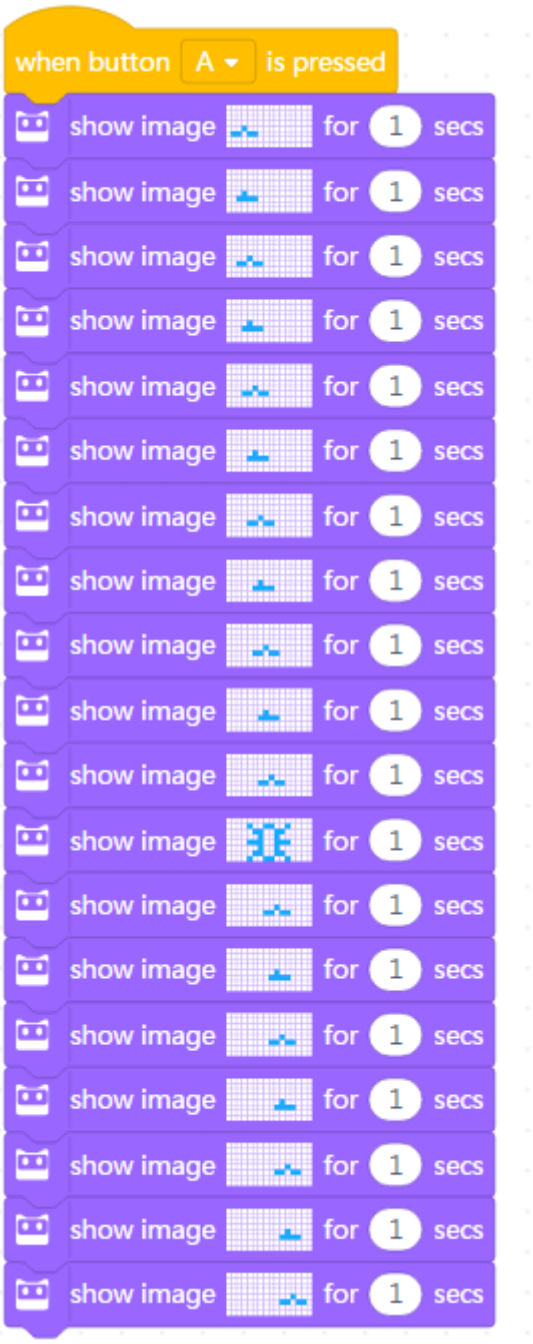
The teacher: Tell students the bomb can't count down and ask them to find the bug in the program.

Students: Try finding the bug in the program.

The teacher: Ask students whether they find the bug or not.

Students: The bug is found. Students need to rearrange the blocks and make sure the blocks are arranged in the correct order.

3. The Earthworm and the Bug



The teacher: Tell students a story: A little earthworm meets a big insect when crawling on the ground. Students need to remove the insect to let the earthworm keep moving forward (replace the insect image with an image of crawling earthworm). Ask students to find out the bug in the program.

Students: Examine the program and try to find the bug.

The teacher: Ask students whether they find the bug or not.

Students: The bug is found. Students need to replace the insect with an image of crawling earthworm.

Have students complete the tasks above in the form of pair programming.

Tell students that there will be a presentation session. Students are expected to share their projects with the whole class and give their answers to the following questions:

1) What is your project about?

2) Did you come across any problems? How did you solve them?

5. Presentation

You can show some well-done projects or have volunteers present their works. When sharing the projects, students are supposed to give their answers to the questions proposed by you.

After one student presents his or her project, you can invite some students to comment on the project (what is good about the project and where to improve). Then give your own comments.

6. Wrap up

You need to give a summary of today's lesson:

Remind students that bugs are inevitable when we write programs. So it's important to know how to identify bugs in programs and how to fix them. In mBlock 5, coding blocks are arranged in the order from top to bottom so we should go through each line of the code in the same order to find out bugs if programs are not running as expected.

7. Students' Self-review

Please find the attached self-review worksheets. Hand out the copies and ask students to spend a few minutes filling the worksheets.

Lesson 17 My Speedway

Lesson Plan

Overview:

Use mBlock 5 to design a simple game.

Teaching Objectives:

1. Know how to use mBlock 5 to design game scenes and characters;
2. Master the ways to achieve dynamic effects
3. Know how to create simple games

Preparation:

1. A whiteboard and a whiteboard marker(or you can use a blackboard and chalks)；
2. One Codey and a Bluetooth dongle (or the USB cable) per student but it's fine if 2 or 3 students share one set;
3. A computer with installed mBlock 5 per student but it's fine if 2 or 3 students share a computer;

Teaching procedure:

1. Review:

Ask students:

1) What did you learn from the previous lesson? What tasks did you complete?

Sample answers:

We learned to use variables and conditionals and completed the task “Rock-Paper-Scissors”

2. Explain New Concepts:

Today we are going to learn to use mBlock 5 to design game characters and scenes.

Navigate students: “When we are new to a game, the first thing we notice about the game is its art design, in other words, how it looks like. The art design is about many things, including main characters, costumes, color contrast, overall style and backgrounds. In most cases, the moment characters and scenes come into sight, we know roughly what the game style is and how we should play the game. Actually, how players play a game is mostly decided by its characters, scenes and back stories. And these elements work together to determine which group of potential players will find the game attractive. In this sense, we have no reason to ignore the importance of characters and scenes designs when we’re talking about a game. When a game has characters and scenes with more intricate details, the game is more visually enjoyable and naturally will keep players engaged.

Then how do we design good characters and scenes?

I. Defining Parameters

1. Defining the tone and scenes based on the game theme (warm tone/cold tone; city/country)

In different backgrounds, characters vary a lot. To be specific, characters could differ from each other regarding their identity and status and the functions of characters are defined by the background where they are placed. From the historical background and the nationality settings of

the level design, players can know the features of the whole game and characters even before they start playing it.

2. Defining the character style and type (modern, cartoon, ancient, futuristic, etc.) based on the game theme and the back story

Cute style: Cartoon characters, bright colors, adorable style. All these elements make the game light-hearted and approachable.

Realistic style: Design the characters based on real persons. The characters and the real persons are similar in sizes, proportions and facial features.

Surreal style: The characters are surreal in sizes, proportions, and shapes. All the characters in this type of games look like unusual creatures.

Geometric style: Characters are designed by piecing together simple geometric figures.

3. Defining the shapes, motions, and accessories of characters based on the relations between characters.

II. Finding Inspirations

1. Design ideas come from life. That is to say, we can get inspirations from the real world's characters and scenes when designing game characters and scenes.

2. Find inspirations in existing game characters and scenes.

3. Find inspirations in other designs from different cultures.

III. Explaining Elements

1. Straight lines suggest peace.

2. Ovals suggest kindness.

3. Sharp triangles suggest speediness, sharpness or evil.

4. A beefy man suggest power.

5. A slender character suggest softness.

（You can make the list longer.）

Tell students to have a close look at the following characters and answer the questions below:

1. Ask students: "What features ddrafo these characters have in common?"

2. Ask students: "What feelings do you have for these characters respectively?"

3. Ask students: "Could you sort out these characters? Who is sweet and who is evil?"

Tips:

In consideration of the copyright issues, all the pictures above are for reference only. You can download the pictures online. But the pictures above will be deleted if the use of these pictures is against the copyright laws.

Give your feedback on the students' answers and direct students to summarize the fundamentals of game character design.

3. Game——Follow the Instructions

1）Brainstorming: Students will understand what kind of characters are suitable for a racing game.

Navigate students to think about one question – What kind of characters and scenes should we design for a racing game?

The teacher: "Kids, we've learned about the basic rules for designing game characters and scenes. Now, we are going to create a racing game. So what kind of characters and scenes should we design? As we all know, car racing is a top-class sport. In the real world, we have the Formula 1 racing that attracts many fans (show videos or pictures here). But few of fans get the chance to experience the sport because it is so dangerous and risky. However, thanks to the rapid development of computer technology, we've got plenty of realistic racing games that allow us to have some fun, for instance, *FI Racing Legends, World Circuit, Need for Speed*."

Have students think about the question and discuss with others – (the teacher saying to students) "There are plenty of racing games available on the Internet. What are their features? If you are to design a racing game of your own, what kind of characters and scenes will you put in your game?"

Typical features of other racing games:

- Simulating real racing cars by making adjustments to their looks
- Cartoon versions of racing cars (bright colors, sweet & cute).
- Geometric pixel characters. These types of characters are relatively easy to design.

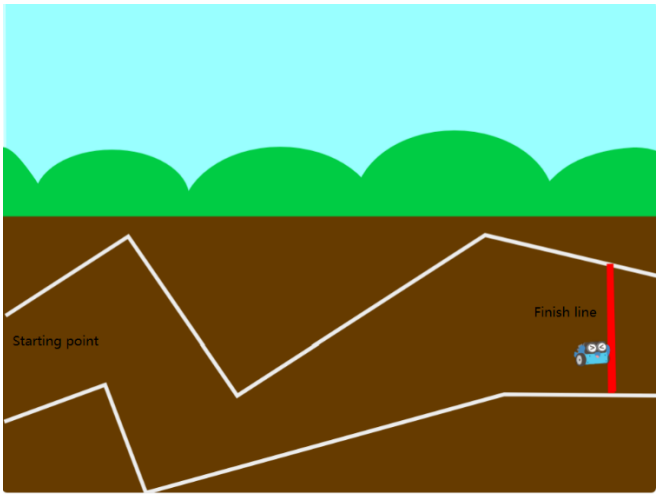
Leave students some time to discuss in groups and have them think about how to design their game characters and scenes.

The teacher summarizes: "I've been thinking about what kind of characters and scenes I should create, too. The theme of today is *Racing Games*, so the main character should be a racing car and the scene should be a race track."

4. Demonstration

1) **Prototyping:** Have students learn how to use mBlock 5 to design the control schemes for their games.

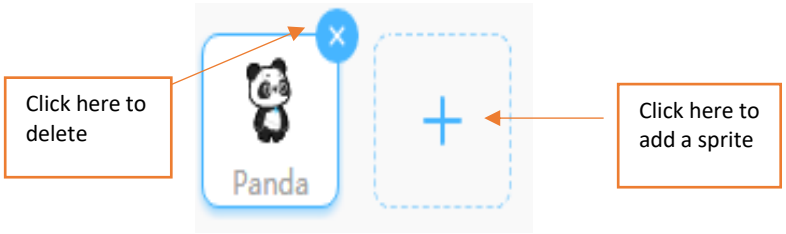
Physical prototype: The teacher will draw a draft to make his or her conceptual design clear to the class. (Check out the slide for details)



The teacher gives students instructions: “Since we are creating an electronic racing game, we have to turn the physical prototype into a digital prototype using mBlock 5.”

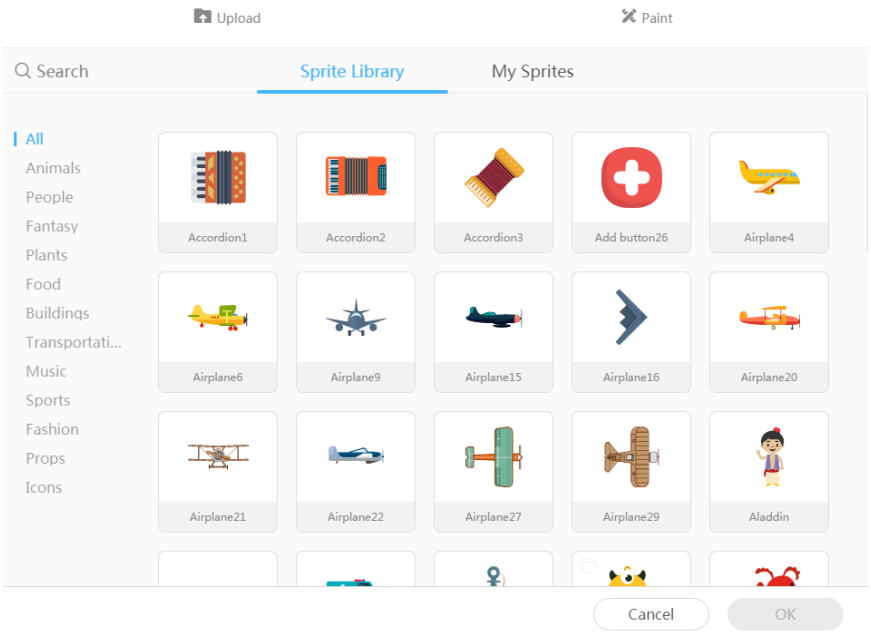
Digital prototype: “After we nail down the character and the scene, next we are going to make it real. Step by step.”

- Delete the sprite "Panda":

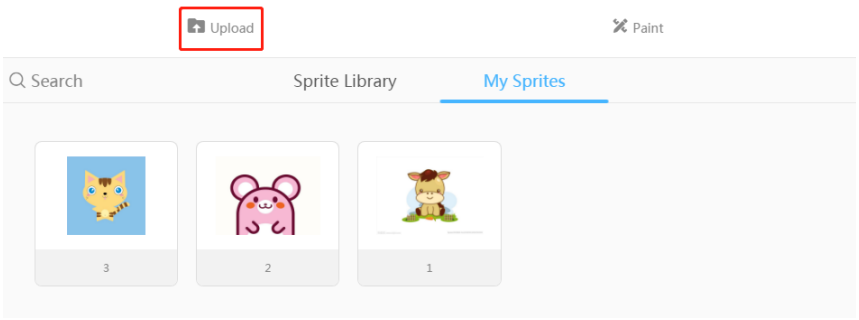


- Add the sprite "racing car". The way to add a sprite:

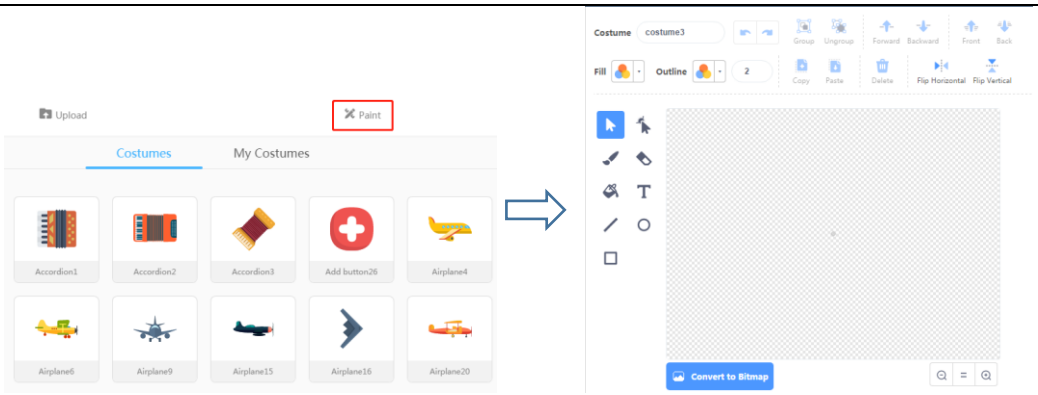
① Pick a sprite directly from the library:



② Search for a picture on the Internet and import the picture:



③ Draw a sprite yourself:



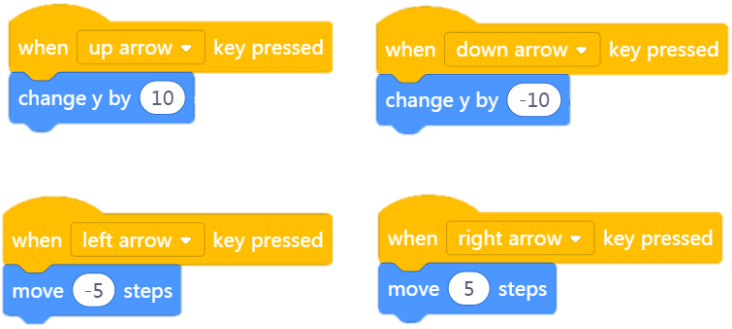
- Use the arrow keys to move the racing car in a specific direction:

① Press the "Up arrow" to move the car upwards;

②Press the "Down arrow" to move the car downwards;

③Press the "Left arrow" to move the car to the left;

④Press the "Right arrow" to move the car to the right.



- Add the background "speedway":

① Pick the background directly from the library;

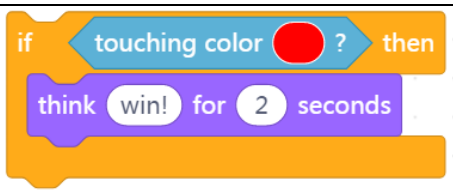
②Search for a picture on the Internet and import it;

③Draw the background yourself.

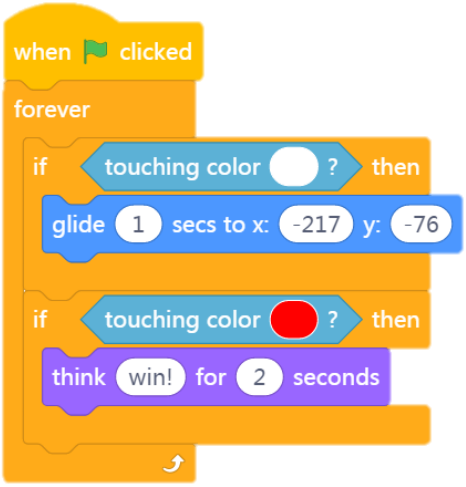
- Enrichment task (Optional):

① Make sure the car drives inside the race track. Once the racing car runs outside the race track, the car returns to the starting place. Figure out how to achieve this effect;

② When the car arrives at the red finish line, the car shows the word "win!".



Sample program:



2) Playtesting: Students will have a better understanding of Playtesting - a pivotal step in game design.

The teacher guides students: "Anyone wants to try this game and share with us what you think of it? Is there anything to improve about the game? Or is there anything that you feel not right?"

3) Reiterating& Implementing: Students will know how to reiterate the prototype and implement it.

Reiterate the prototype based on the feedback from students (players).

5. Practice

Assignment: *Racing Car*

1) Selecting a solution: "We've just discussed how to design characters and scenes for a racing game. And you came up with so many ideas. But now you have to pick out a best one among your own ideas. After that, follow the conceptual idea to design your game, making the racing car running."

2) Prototyping: "Draw a draft or create a prototype based on your idea. Or you can use mBlock 5 to write programs straightaway."

3) Playtesting: "When your game is ready, put your hands up. You can invite some target players to experience the game. Of course, you can invite me to try it."

4) Reiterating & Implementing: "Fix bugs based on the players' feedback. Perfect your game."

6. Presentation

Allow each student to share his or her game with the class. At the end of this session, let students vote for the best design of the day.

7. Wrap up

Game design process: brainstorming-prototyping-playtesting-iterating & implementing. Today we learned to use mBlock 5 to design a game, and applied our programming knowledge to completing the task.

8. Students' Self-review

Please find the attached self-review worksheets. Hand out the copies and ask students to spend a few minutes filling the worksheets.

Lesson 18 Game Control Schemes

Lesson Plan

Overview:

Use mBlock 5 to design an interactive scheme for a game.

Teaching Objectives:

1. Know how to design an interactive scheme for a game.

Preparation:

1. A whiteboard and a whiteboard marker(or you can use a blackboard and chalks)；

2. One Codey and a Bluetooth dongle (or the USB cable) per student but it's fine if 2 or 3 students share one set;

3. A computer with installed mBlock 5 per student but it's fine if 2 or 3 students share a computer;

Teaching procedure:

1. Review:

Ask students：

1) What did you learn from the previous lesson? What tasks did you complete?

Sample answers:

1) We learned how to add new sprites and design a simple game.

2. Explain New Concepts:

Today we are going to learn how to design a game control scheme.

When we design a game, the first thing we consider is choosing a proper gaming controller for the game. It’s because how we control the game can vary a lot across different controllers. And we control games mainly using mice, keyboards, controllers (PSP, PS, Wii Remote or Nintendo D-pad) or VR headsets. In most cases, players can choose to use these controls either collaboratively or separately.

Different types of games have their specific control schemes. The following chart shows us the pros and cons of using a controller VS using a mouse + a keyboard.

	A controller	**A mouse + a keyboard**
Good stuff	Holding a controller in hands feels good; Easy to operate; Provide a good immersive experience; No gesture constraints.	Precise; Speedy; Suitable for PC games.
Bad stuff	Poor accuracy; Extra expenditure	The gaming experience might be poor; Need a PC to work with
game genres	Action, Fighting, Driving	Shooter, Real-time strategy