makeblock

Lesson 4: Game Conflicts

Teachers: Seven & Claire	**Grades:** 3-5	**Time required:**90 min
Subject: Game Design	**Teaching objective:** **Students will** 1) Develop the skills to design complex game mechanics - Game Conflicts. 2) Know how to spice up their racing games by increasing the difficulties.	
Teaching Materials		**New Concepts**
Teacher: Codey, a computer with installed mBlock 5, lesson plans, and the slide	**Student:** Codey, computers with installed mBlock 5	Coding Block: Sensing, Operators
The Teaching Procedure		

Introduction: The teacher will give students an overview of today's lesson – Game Conflicts. (Objective)Students will understand the concept of "Game Conflicts" and master the know-how to design game conflicts.

1) The teacher gives an introduction first: "In the previous lesson, we've learned how to design game rules. For example, we set a rule for the game: The racing car should not run outside the track. The rules settings make the racing game complete."

2) **(The teacher saying to students)**: "In today's lesson, we are going to learn about how to make our games more appealing. To this end, we'll have to increase the difficulties of the game by giving the game complicated rules. Complicated rules spice up games, thusly stimulating players

to go the extra mile to win the games. In contrast, simple rules make the game less challenging. Players can crack the tasks so easily that they can hardly get the sense of achievement out of participation."

3) We have a term that describes the behavior of increasing the difficulties of a game - **Game Conflicts**. By designing challenges for games, we try to make it hard for players to accomplish the goals of games directly. There are three types of game conflicts:

Obstacles: Design obstacles in games. The types of obstacles could be categorized into fixed obstacles and moving obstacles

Opponents: Add opponents in games (obstacles or other players in multiplayer games).

Dilemmas: Players might come across multiple solutions. But each solution comes with pros and cons.

Guided Practice:

1) Brainstorming: (Objective) Have students consider what conflicts fit best in racing genres.

① **(The teacher saying to students):** "In today's lesson, we'll learn how to make our game more appealing. So, we will move on to designing games conflicts in the game. Considering the input mode of the racing game (only one Codey can be connected to one computer so we can't add opponents), the suitable game conflicts will come from obstacles or dilemma-based choices. What kinds of game conflicts will you design for your games? Try to figure out the question. Don't make the conflicts too challenging. Otherwise, players will quickly give up the game when they fail to accomplish the goals. "

The teacher should join students for discussions and write down students' ideas on the blackboard. Give students enough time to discuss and work out the solutions. Don't forget to remind students of one important thing: take a playercentric approach to designing the game conflicts.

The teacher then tells students: “ All of your ideas are brilliant! And each of your conflict settings is unique, making your game different from each other. I’ve also come up with some game conflicts. I’ve been thinking about designing moving obstacles in the game. The position where the moving obstacle appears changes each time. When driving forwards, the racing car has to avoid those obstacles all along. Once touching obstacles, the car has to go back to the place where it sets out and restarts the game.”

②The teacher asks students: “ What moving obstacles do you think the car will come across while driving?”

The teacher needs to join students for discussion and write down students’ ideas on the blackboard. Organize students to have a discussion on those ideas.

In the end, the teacher should give his or her own design idea: using bats as moving obstacles in the game. The inspiration of using bats as moving obstacles comes from a report: Near those building with glass walls, people could always find bats which are already dead. Besides, as shown in the researches by German scientists, although bats has the echolocation that enables them to fly without any trouble in the dark, the location system won’t work when running into slippery and vertical surfaces. In this case, bats might hit the glass windows or glass walls. So surely there is a high chance that bats will hit the glass windows of the racing car. In consideration of this, the teacher decides to set a bat as the moving obstacle in the racing game. The bat will keep falling down from the sky randomly. If the bat hits the window of the car, there will be a remark left on the window, which makes hard for the car to keep driving. Therefore, the rule is designed to be like this: If the racing car hits the bat, the car will go back to the place where it sets out (after the glass is cleaned and restarts.)

2) Prototyping: (objective) The students will know how to design game conflicts using mBlock5.

Physical prototype: The teacher draws a bat as the draft.

Bat 1: A complete bat;

Bat 2: A bat smashed to pieces by the racing car. Use black spots to represent the smashed bat;

Digital prototype: Use mBlock 5 to code the bats. First add the sprite “bat” to the stage.

Game conflict:

The racing car has to avoid the bats falling from the sky when driving.

Outcomes:

① The game will be over once the bat hits the bat;

② The game continues if the car doesn’t hit the bat. But the bat will appear again later.

Questions:

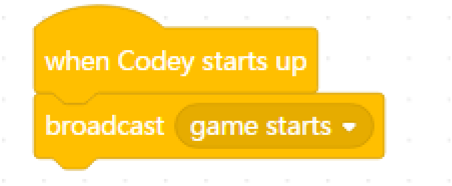
① How do we keep the bat falling from the sky?

② How do we write programs to ensure that the bat explodes once it hits the car? (switch the sprites)

③ How do we make the bat appear in the sky again after it hits the ground?

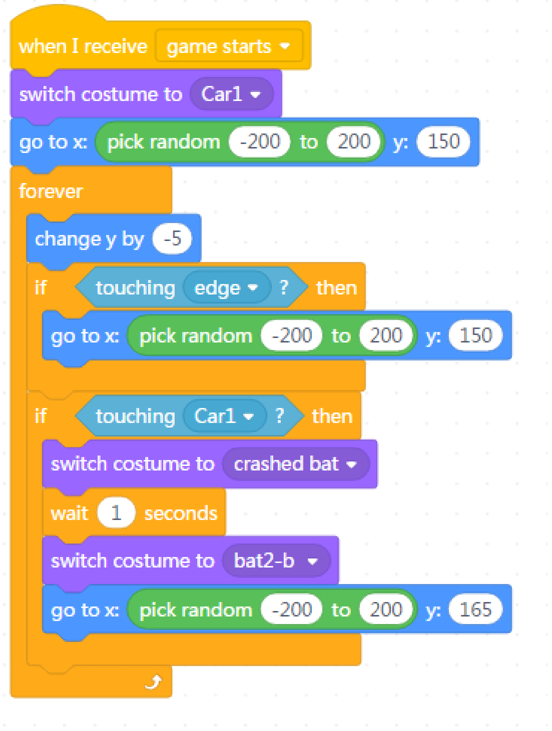
④ How do we make the bat come out of a random place each time?

The program for controlling Codey:



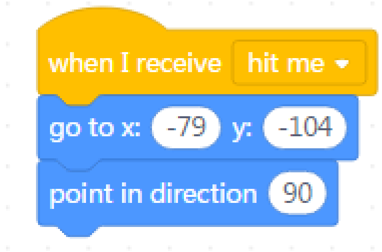
```
when Codey starts up
broadcast game starts ▾
```

The program for controlling the bat:



```
when I receive [game starts ▾]
switch costume to [Car1 ▾]
go to x: (pick random (-200) to (200)) y: (150)
forever
    change y by (-5)
    if <touching [edge ▾] ?> then
        go to x: (pick random (-200) to (200)) y: (150)
    if <touching [Car1 ▾] ?> then
        switch costume to [crashed bat ▾]
        wait (1) seconds
        switch costume to [bat2-b ▾]
        go to x: (pick random (-200) to (200)) y: (165)
```

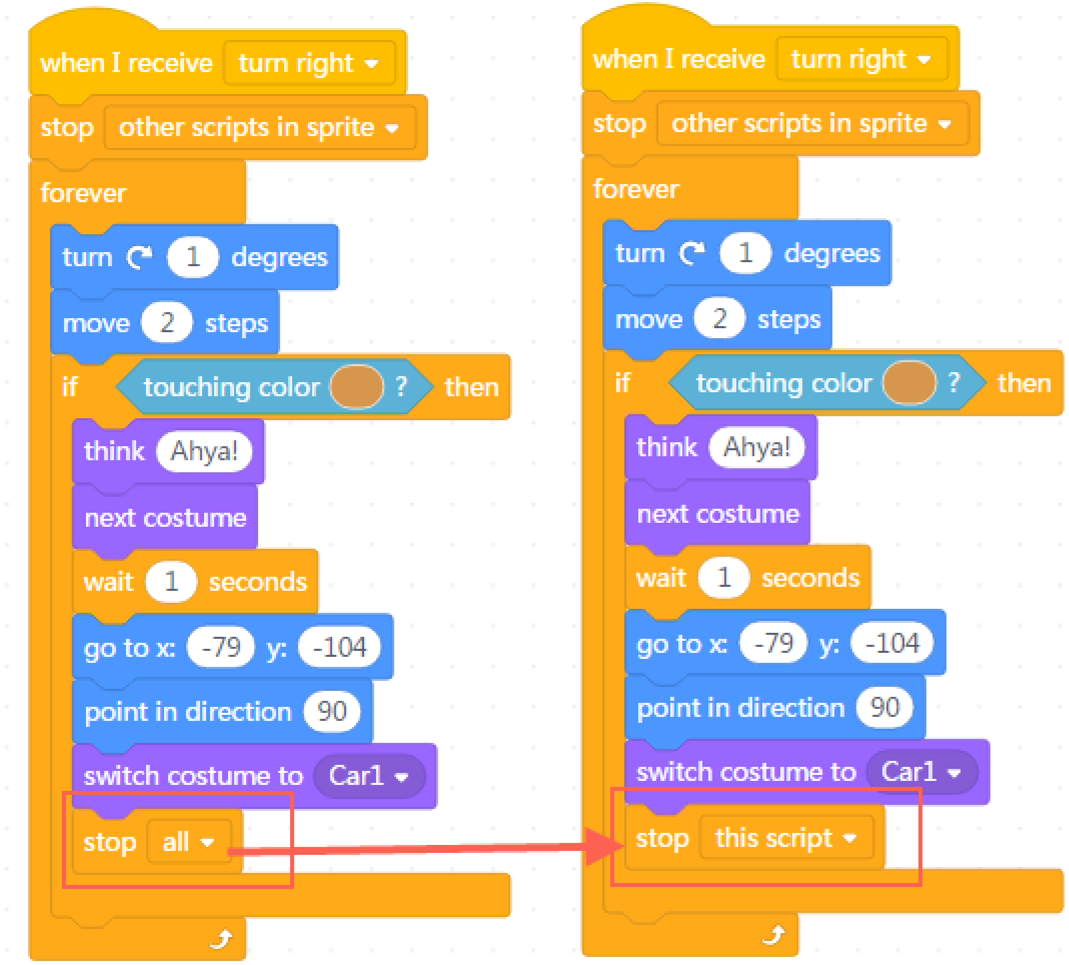
The program for controlling the racing car:



```
when I receive [hit me ▾]
go to x: (-79) y: (-104)
point in direction (90)
```

What’s the problem: Why does the bat come to a halt all of a sudden?

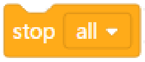
Identify the issue: The program for controlling the racing car needs revising:



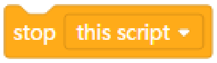
```
when I receive [turn right ▾]
stop [other scripts in sprite ▾]
forever
    turn ↻ (1) degrees
    move (2) steps
    if <touching color [●] ?> then
        think (Ahya!)
        next costume
        wait (1) seconds
        go to x: (-79) y: (-104)
        point in direction (90)
        switch costume to [Car1 ▾]
        stop [all ▾]
```

```
when I receive [turn right ▾]
stop [other scripts in sprite ▾]
forever
    turn ↻ (1) degrees
    move (2) steps
    if <touching color [●] ?> then
        think (Ahya!)
        next costume
        wait (1) seconds
        go to x: (-79) y: (-104)
        point in direction (90)
        switch costume to [Car1 ▾]
        stop [this script ▾]
```

What's the difference between the two blocks:



```
stop [all ▾]
```



```
stop [this script ▾]
```

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

The teacher asks students: "Anyone wants to experience this game and shares with us how you feel about the game? Is there anything that you think should improve?"

4) Iterating & Implementing: (Objective) Students will know how to reiterate the game prototype and implement the game.

The teacher will make modifications based on the feedback from students.

Independent Practice:

It’s Your Turn!

1) **Selecting a solution: (The teacher saying to students)** “We’ve just discussed how to make your racing game more engaging. You came up with so many ideas, but now you have to narrow down the ideas to only one. After that, you can start creating your game.”

2) Prototyping: (The teacher saying to students) “Add sprites to your game based on your design concept. Then starts coding the sprites.”

3) Playtesting: (The teacher saying to students) “When your game is ready, put your hands up. You can invite target players to playtest your prototype. Of course, you can invite me to try it.”

4) Iterating & Implementing: (The teacher saying to students) “Fix bugs based on the players’ feedback. Try everything to perfect your game.”

Share:

Presentation:

Encourage each student to share his or her own game with the class and tell them to invite classmates to experience their game.

At the end of this session, let students vote for the Best Game of the Day.

Differentiation & Modification

For advanced students	**For struggling students**	**Strategies to maximize engagement**

1)Add a new moving obstacle (gold coin). The racing car should capture the gold coin before the coin falls on the ground and tries to avoid the bats at the same time. 2)Add a new track shortcut to complicate the dilemma-based choices. But make this track narrower.	1)Make the game conflicts simpler, like reducing the number of bats or increasing the intervals.	Have students vote for the Best Game of the Day
Comments:		
Teachers’ Reflections:		