

The logo for AlbertaSat features a stylized satellite or orbital path icon. It consists of two white elliptical orbits intersecting at a central point, with a green line representing a satellite or a specific orbital path. The text "AlbertaSat" is written in a clean, white, sans-serif font to the right of the icon.

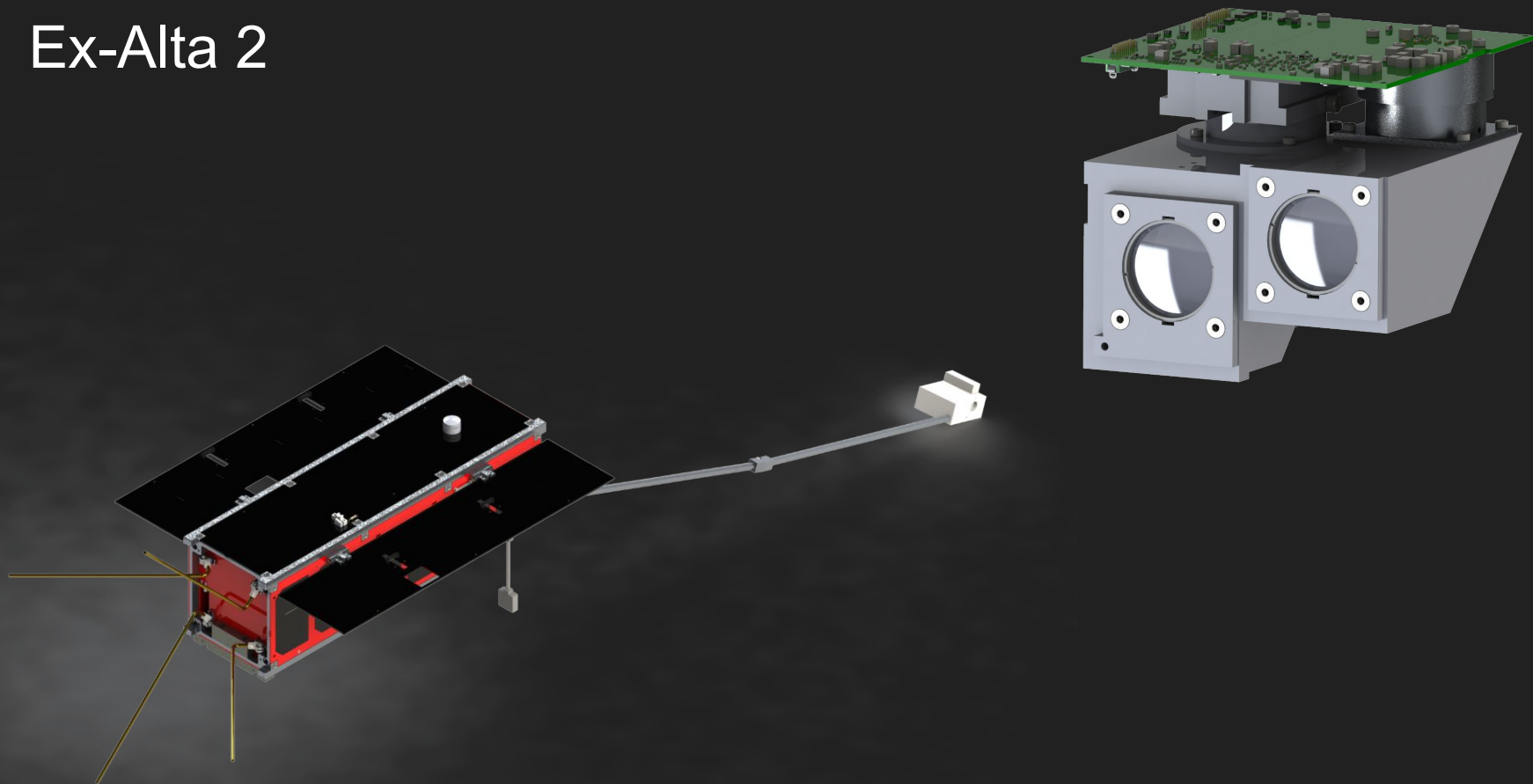
AlbertaSat



Ex-Altá 1



Ex-Altá 2



```

C main.c 3 C sTransm.c 4
C sTransm.c > add_vBuffer(int)
27 // * Simulated registers so that we can "store" values
28 static uint8_t reg0 = 0, reg1 = 0, reg3 = 0, reg4 = 0, reg5 = 1;
29 static uint8_t reg17 = 30, reg18 = 3, reg19 = 1, reg20 = 0;
30 static uint8_t reg21 = 0, reg22 = 0, reg23 = 0, reg24 = 0;
31 static uint8_t reg25 = 0, reg26 = 8, reg27 = 8, reg28 = 8;
32 static uint8_t reg29 = 8, reg30 = 50, reg31 = 0, reg32 = 255;
33 static uint8_t reg33 = 192, reg34 = 8, reg35 = 8, reg36 = 16;
34 static uint8_t reg37 = 16, reg38 = 192, reg39 = 64, reg40 = 0, reg41 = 9;
35
36 /**
37  * @brief
38  * Adds to the virtual Buffer
39  * @de
40  *
41  * @ever
42  * @at
43  *
44  * @pa
45  * The number of bytes to add to the buffer
46  * @return STX_return
47  * Success of the function defined in sTransmitter.h
48  */
49 STX_return add_vBuffer(int n_bytes) // Replace with spi_writeData eventually
50 {
51     spi_writeData_Expect();
52     spi_writeData();
53
54     for(int j = 0; j < n_bytes; j++){
55
56         // Time Delay
57         // sleep(S_DATA_TIME);
58
59         // Overrun?

```

Coding with AlbertaSat

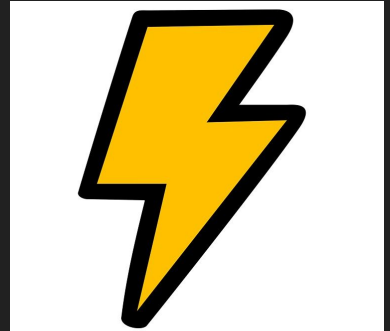
```

66     }
67     reg19 = 0;
68     continue;
69 }
70
71
72     if(reg25 == 255){
73         reg24++;
74         reg25 = 0;
75     }else{
76         reg25++;
77         uint16_t b_count = 0;
78         STX_getBuffer(S_BUFFER_COUNT, &b_count); // TR register update
79         if(b_count > 2560){reg19 = 0;}
80     }

```

Why is coding important for satellites?

- Communication
 - Between mission control and the satellite
 - Between subsystems in the satellite
- Power regulation
- Control of the payload



Scratch is a coding language that uses blocks to perform programming

- These blocks come in different shapes and colors
- The blocks can be arranged to execute different functions
- They are connected together like puzzle pieces



The game you'll make: Asteroid Dodger!



Motion Blocks

- These blocks control the motion of the objects
- They are indicated in **blue**



Control Blocks

- These blocks use logic to perform certain tasks
- These blocks use 'If' statements:
 - **If ___ happens, then do ___**
- These blocks do loops:
 - **Repeat until ___**
- These blocks are indicated in **orange**



Sensing Blocks



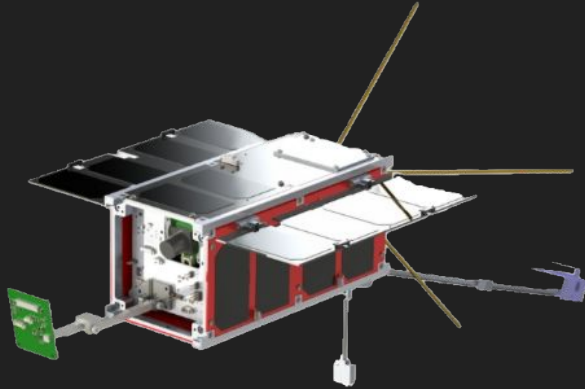
- The sensing blocks 'listen for' certain events
- These are indicated in **light blue**.



Looks Blocks



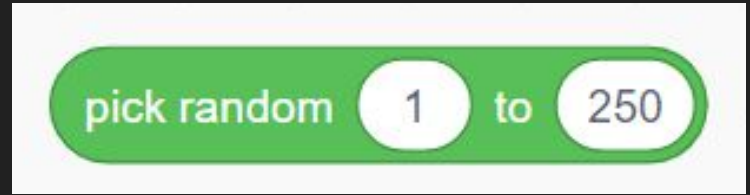
- These blocks change the appearance of the sprite
- The appearance of the sprites are called **costumes**
- They are indicated in **purple**



Operator Blocks

- These blocks do things regarding:
 - Math
 - Logic
- They are indicated in **green**

- We'll be using an **Operator** block to randomly generate where the asteroids will come from!



The Main Programming Window:

The image displays the Scratch programming environment. The top navigation bar includes the Scratch logo, a globe icon, and menu options: File, Edit, and Tutorials. On the right side of the top bar, there are links for "Join Scratch" and "Sign in".

The left sidebar contains a vertical menu of block categories: Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, and My Blocks. The "Motion" category is currently selected, showing a list of motion-related blocks such as "move 10 steps", "turn 15 degrees", "go to random position", "glide 1 secs to random position", "point in direction 90", "change x by 10", "set x to -142", "change y by 10", "set y to -1", "if on edge bounce", and "set rotation style left-right".

The main workspace is a large grid where code blocks are placed. A green rectangular border highlights a specific script. This script begins with a yellow "when clicked" block. It is followed by a sequence of blue motion blocks: "go to x: -160 y: 0", "change y by 10", "change x by -10", and "change y by -10".

Following these motion blocks are three orange "forever" loops. Each loop contains an "if" block with a "then" block. The "if" blocks are connected to "key up arrow pressed?", "touching asteroid?", and "touching asteroid2?" blocks. The "then" blocks are connected to "key down arrow pressed?", "key right arrow pressed?", and "key left arrow pressed?" blocks. At the end of the script, there are two orange "stop" blocks, both set to "all".

On the right side of the workspace, there is a stage area showing a satellite sprite named "AuroraSat_sprite" with a position of x: -142 and y: -1. Below the stage, there is a "Backdrops" area with two asteroid sprites named "asteroid" and "asteroid2".

Sprite/Character Window:

The image shows the Scratch IDE interface. The main workspace contains a script for a sprite with the following blocks:

- when green flag clicked
- go to x: -160 y: 0
- if on edge, bounce
- key up arrow pressed?
- key down arrow pressed?
- when green flag clicked
- change y by 10
- change x by -10
- touching asteroid?
- key right arrow pressed?
- when green flag clicked
- change y by -10
- change x by -10
- touching asteroid2?
- key left arrow pressed?
- forever loop containing:
 - if then
 - if then
 - if then
- forever loop containing:
 - if then
 - if then
 - if then
- forever loop containing:
 - stop all
 - stop all
- point in direction 90
- point towards mouse-pointer
- change x by 10
- set x to -142
- change y by 10
- set y to -1
- if on edge, bounce
- set rotation style left-right
- x position

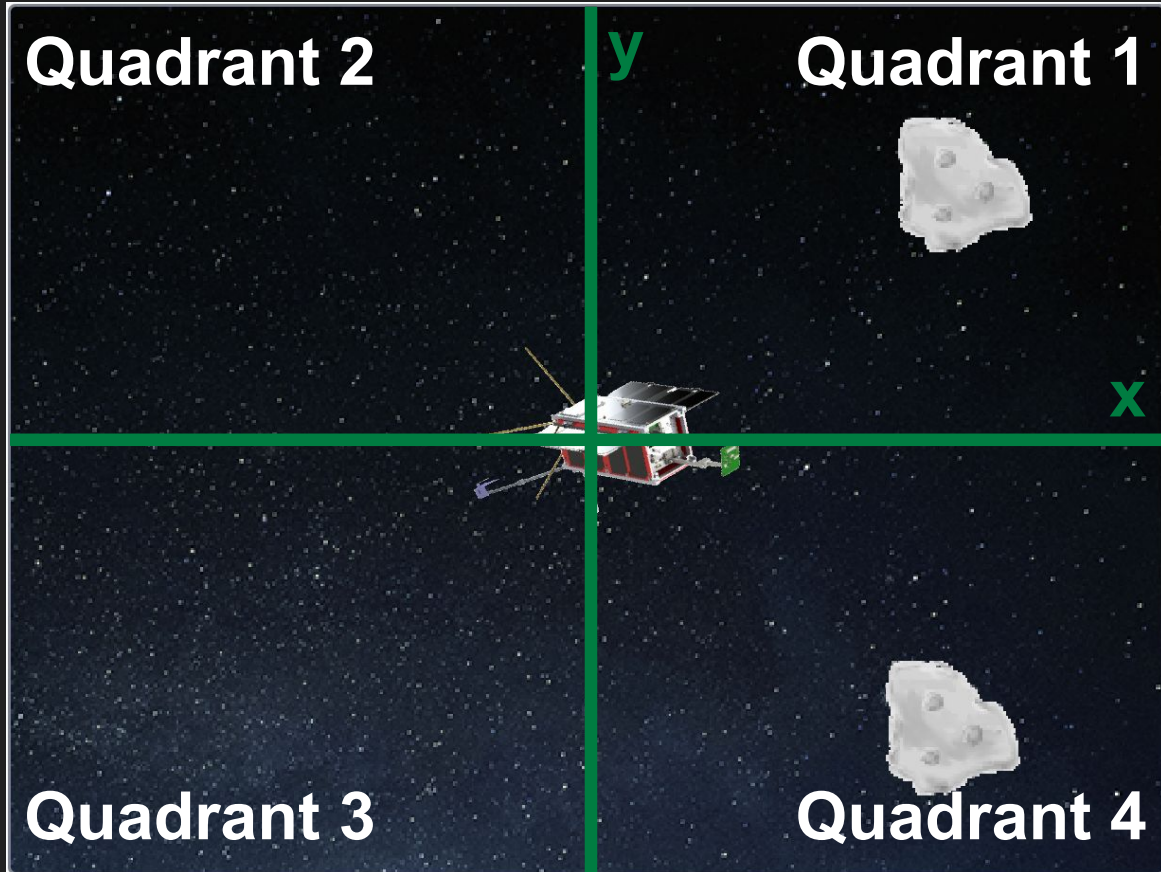
The right-hand side of the interface shows the Sprite window. The sprite is named "AuroraSat_sprite" and is currently at x: -142 and y: -1. The sprite window displays three sprites: "AuroraSat...", "asteroid", and "asteroid2". The "AuroraSat..." sprite is highlighted with a green box. The Stage window shows a satellite in space with two asteroids.

Game Preview Window:

The image displays the Scratch IDE interface. At the top, the menu bar includes "Scratch", "File", "Edit", and "Tutorials". Below the menu bar, there are tabs for "Code", "Costumes", and "Sounds". The left sidebar contains a category menu with icons for Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, and My Blocks. The main workspace is filled with a grid of code blocks. The "Motion" category is selected, showing various blocks like "move 10 steps", "turn 15 degrees", "go to random position", "glide 1 secs to random position", "point in direction 90", "change x by 10", "set x to -142", "change y by 10", "set y to -1", "if on edge bounce", and "set rotation style left-right". The "Control" category shows "when green flag clicked" and "forever" loops. The "Sensing" category shows "if on edge bounce" and "touching asteroid?". The "Operators" category shows "if then" blocks. The "Variables" category shows "stop all" blocks. The "My Blocks" category shows "x position".

At the top right of the IDE, there are two icons: a green flag and a red circle. Arrows point from these icons to a preview window on the right side of the screen. The preview window is framed in green and shows a game scene with a satellite and two asteroids in space. Below the preview window, the "Sprite" panel shows the "AuroraSat_sprite" with its position (x: -142, y: -1) and direction (78). The "Stage" panel shows the "Backdrops" list with "AuroraSat..." and "asteroid2".

The game is a cartesian plane

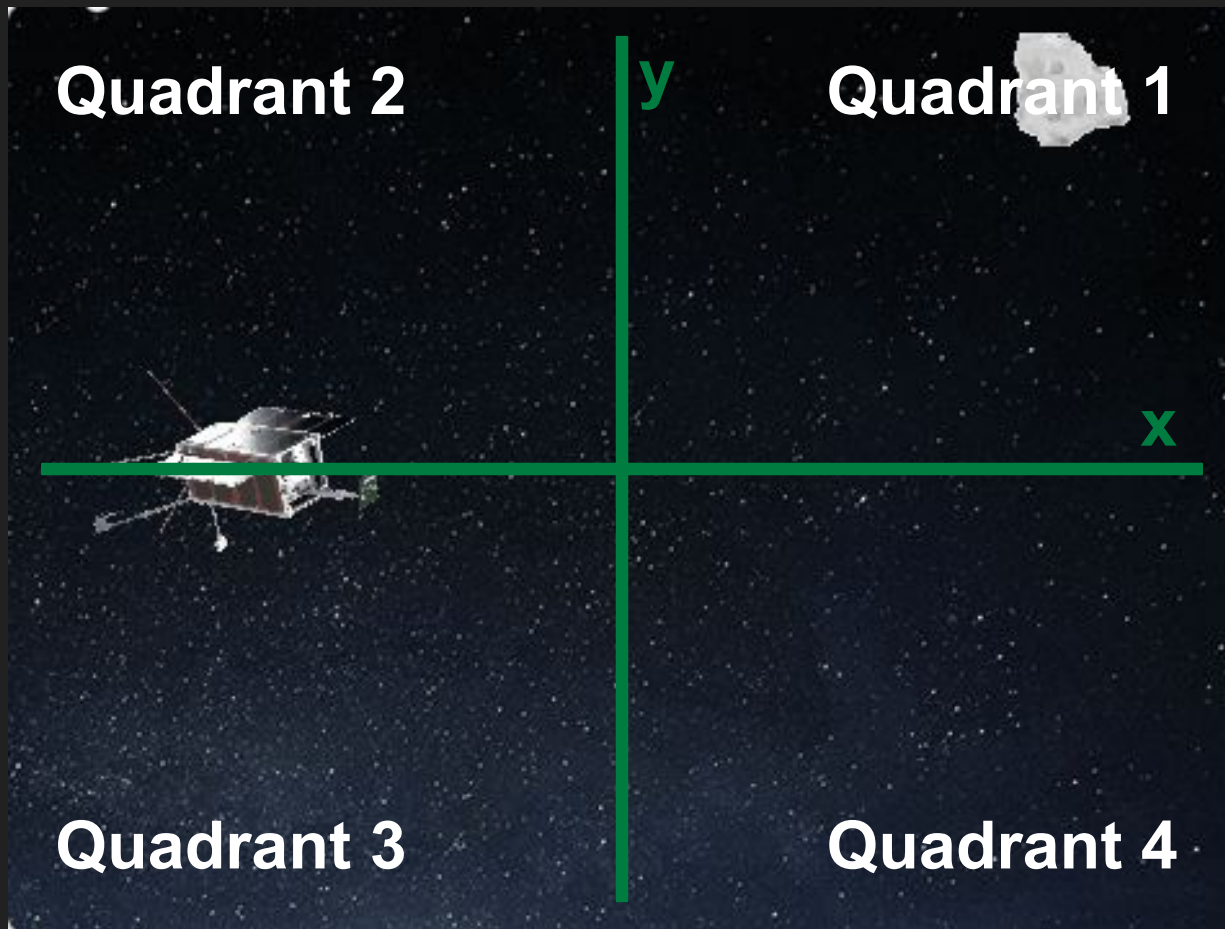


Can you identify the location of the satellite sprite?

(0,0)

If we move the satellite sprite to (-50,50), which quadrant will it be in?

Quadrant 2



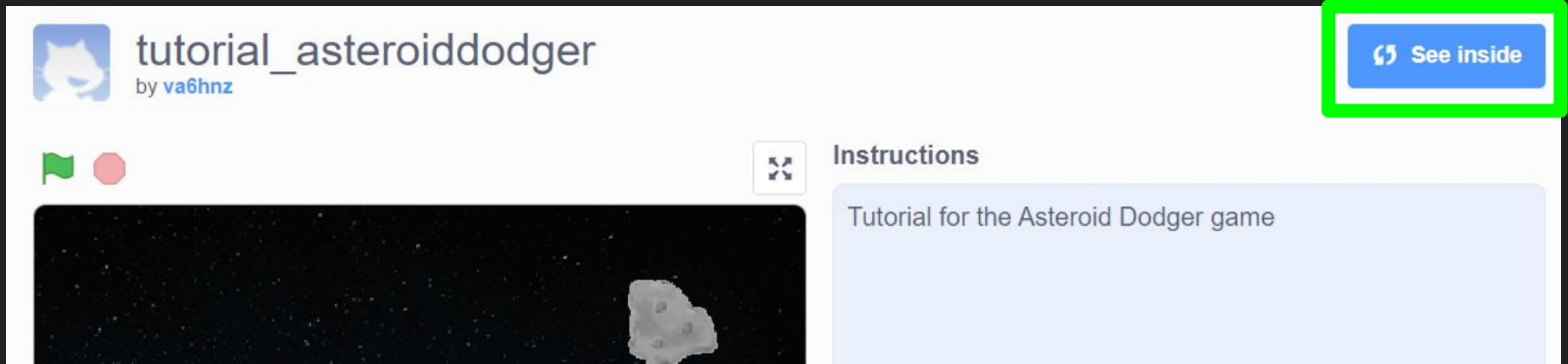
Over to scratch!



Go to the tutorial on the scratch website:

<https://scratch.mit.edu/projects/605478607>

and click on the “see inside” button:



The screenshot shows the Scratch project page for 'tutorial_asteroiddodger' by 'va6hnz'. The page features a video player showing a space scene with a grey asteroid. To the right of the video player is an 'Instructions' section with the text 'Tutorial for the Asteroid Dodger game'. A blue button labeled 'See inside' is highlighted with a red box in the top right corner of the page.


Your screen should now look like this!

The screenshot displays the Scratch IDE interface with a completed script for a satellite game. The script is organized into several sections:


- Initialization:** Starts with a 'when green flag clicked' block, followed by 'go to random position', 'glide 1 secs to random position', and 'point in direction 90'.
- Control:** A 'forever' loop containing:
 - 'point towards mouse-pointer'
 - 'change x by 10'
 - 'set x to -142'
 - 'change y by 10'
 - 'set y to -1'
 - 'if on edge bounce'
 - 'set rotation style left-right'
- Collision Detection:** A 'when green flag clicked' block followed by a 'forever' loop:
 - 'if touching asteroid?' - if true, 'go to x -160 y 0', 'change y by 10', 'change x by -10', 'if on edge bounce', 'key up arrow pressed?', 'key down arrow pressed?', 'touching asteroid?', 'key right arrow pressed?', 'key left arrow pressed?', 'touching asteroid2?', 'if touching asteroid2?'.
 - 'if touching asteroid2?' - if true, 'go to x -160 y 0', 'change y by -10', 'change x by -10', 'if on edge bounce', 'key up arrow pressed?', 'key down arrow pressed?', 'touching asteroid?', 'key right arrow pressed?', 'key left arrow pressed?', 'touching asteroid2?'.
- Termination:** A 'forever' loop containing 'stop all'.

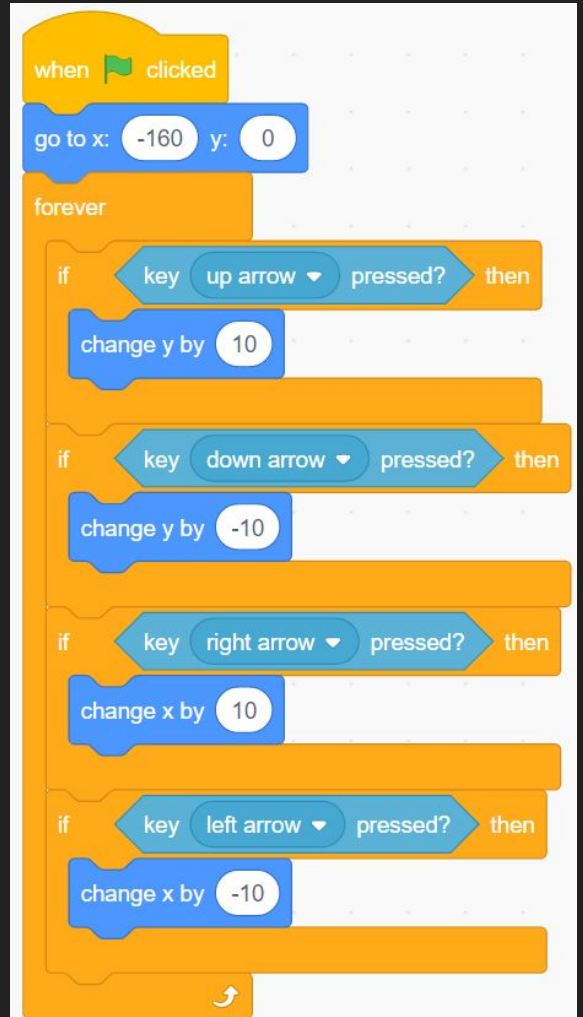
The right-hand side of the IDE shows the stage with a satellite sprite and two asteroid sprites. The sprite panel includes 'AuroraSat_sprite', 'asteroid', and 'asteroid2'. The stage properties show the satellite at x: -142, y: -1, size: 50, and direction: 78.

Programming the AuroraSat_sprite

- You'll notice that when you press  nothing happens just yet!
- Click on the AuroraSat_sprite from the Sprite window:



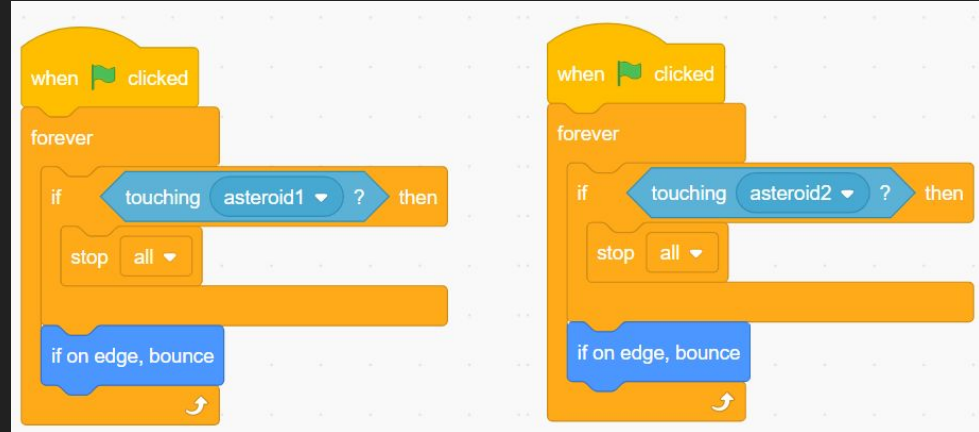
- You should now have all the blocks you need to program the AuroraSAT in the main window!
- First, let's try to get it to move with your arrow keys!
- Try and recreate the picture on the right with your blocks!
- When you think you've finished, click  and try and see if you can use your arrow keys to move the AuroraSat!



What happens when the AuroraSAT runs into an asteroid?!

We have to program the game to stop when the AuroraSAT runs into an asteroid!

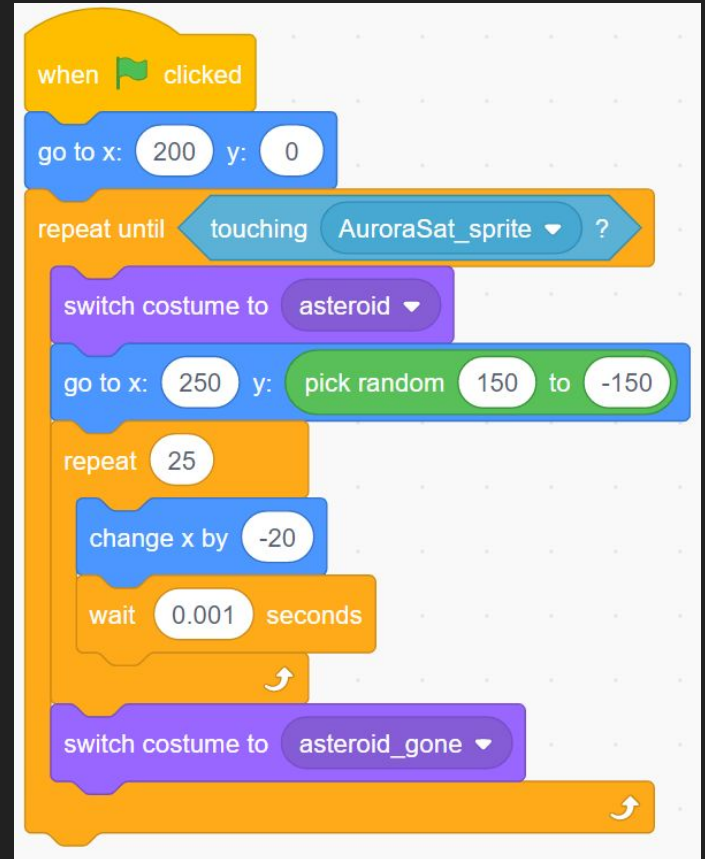
Try and recreate these conditions from your remaining blocks!



Programming asteroid1

You may have noticed your asteroids don't move yet!

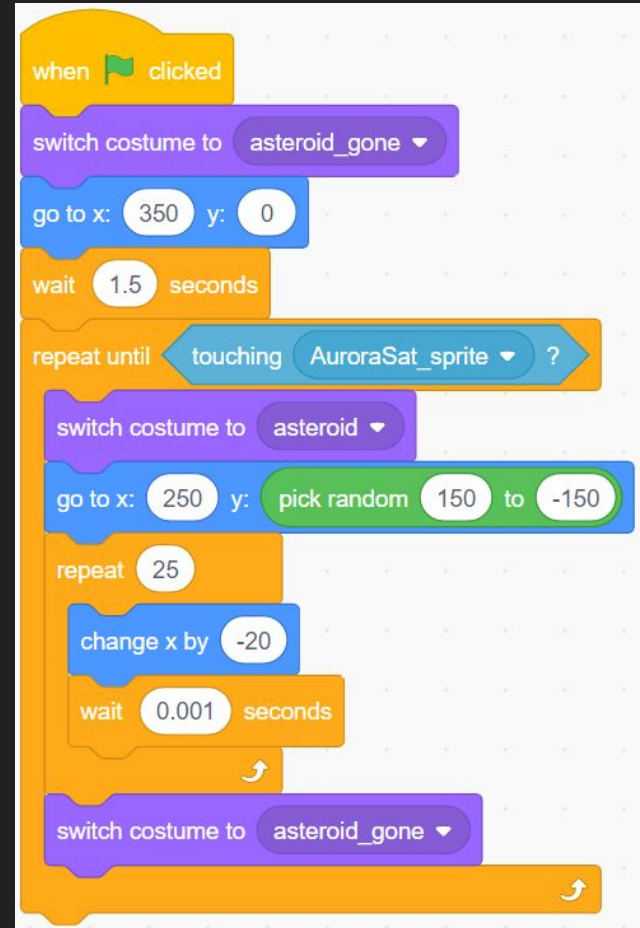
- Select the **asteroid1** sprite from the Sprite window
 - The blocks necessary to program **asteroid1** should now be in the main window!
- Try and recreate the image on the right with your blocks!



Programming asteroid2

Let's add another asteroid!

- Select the **asteroid2** sprite from the Sprite window
- The coding for this sprite has mostly been done, they just need to be connected together!
 - There's a hint if you're stuck!
- The finished blocks should look like this:



Completed Game

<https://scratch.mit.edu/projects/605516110>