

## Practical Session 2: Game Structure and Sprites in Scratch

### Overview

In this session we are going to look at some basic game mechanics, including: Sprites; User input; Backgrounds; Collision detection; and Game structure. You will need to download the “ScratchPygame Resources” from the class materials wiki page.

### Task 2.1

The functionality for Scratch is largely built around Sprite objects. This fits in very well with game development as “Sprite” is generally used to describe a graphical object that can be moved around a game surface.

In this first task, we are going to create a Sprite using the “ship.png” file from the downloaded resources, in a new Scratch project (start a new one now). There are two easy ways of doing this in Scratch:

1. There are three buttons near the bottom-right corner of the Scratch interface. Clicking the middle one “Choose new sprite from file” will allow you to browse the file system for where the file is located, and will create a new Sprite with this image as its first costume.
2. Dragging and dropping an image file from the file system onto the Sprite area of scratch (bottom-right area) will add the image as a new costume for the currently selected sprite, or background.

Once you have the ship sprite created (you should rename the sprite as ‘Ship’ for easy reference), it’s time to get it moving. To do this, we are going to add user-input for rotating the ship and moving it forwards and backwards. Add in the following scripts:

1. When left arrow key pressed, turn anti-clockwise 45 degrees.
2. When right arrow key pressed, turn clockwise 45 degrees.
3. When up arrow key pressed, move forward 10 steps.
4. When down arrow key pressed, move backwards 10 steps.

If you’re unsure on how to do this, refer to the Scratch reference guide, browse through the available commands, or ask a tutor if really stuck.

## **Task 2.2**

A spaceship moving about on a plain white background doesn't look very good, so let's add in a background. The file "starfield.png" in the downloaded resources should do for this.

As with creating sprites from images, there are two easy ways of using an image for a background:

1. Select the stage and view the current backgrounds. You should now be able to select the "Import" button at the top so you can browse the file system for the background file.
2. With the stage selected, drag and drop the image from the file system to either the sprite area or the backgrounds area – this will set the current background to the imported image.

## **Task 2.3**

The pilot of the spaceship has nothing to do, so let's give him some work. His task is to pick-up a strange looking object represented by the image "collect.png". You should create a new sprite named "Pick-up" using the image file.

There are a few in-built ways to detect collisions in Scratch, some that work better than others depending on the situation. For what we need, we can't use the colours black or white to determine if the ship is colliding with the object, but we could use the light grey colour. However, the simplest way to do this would be add a script to the Pick-up sprite as follows:

1. When the green flag is clicked
2. Forever
  - 2.1. If touching Ship
    - 2.1.1. Hide
    - 2.1.2. Stop Script

Notice that I have used outline numbering above to indicate that that the "if" is inside the "forever", and the last two statements are inside the "if". Ask a tutor for clarification if you're unsure of this.

## **Task 2.4**

As it stands, we have a small demo that illustrates sprites, user input, backgrounds and collision detection. The main problem we have just now is that it can only be “played” once as the pick-up disappears once it has been collided with. This problem can be fixed easily by using the Show command, but let’s think about this in terms of game structure:

### Initialisation

When the game starts up, everything should be initialised. This includes hiding sprites you don’t want seen, showing sprites you do want seen, positioning sprites at their start position ...

For this project, a typical initialisation would be something like:

1. Set Stage background to starfield
2. Position Ship at (0,0)
3. Point Ship in direction Right
4. Position Pick-up at position (100,100)
5. Show Pick-up

By adding these initialisation steps at the start (e.g. when green flag clicked), each time the game is started will be the same. This is important when creating games as you want all your users to have a similar experience, unless you intentionally create dynamic environments that change with each play, but this is a slightly more advanced topic.

### Updating (Game loop)

Unlike with most programming languages, there is no need for an explicit game loop in Scratch. Typically, the game loop is processed repeatedly, updating all sprites, user input, game events etc. according to the desired frame rate. In Scratch, each script is self-contained, so you decide if you want them to run once, run when an event happens, or run continuously (using Forever). It is important to note that this can lead to things happening in a different order than you had intended, so keep an eye out for this happening.

### Rendering

It is common that inside the game loop, after updating, the current sprites and background would be rendered to the screen. Again, in Scratch, there is no need to do this explicitly as it is handled for you. All you have to do is position your sprites, and Show or Hide them as appropriate.

**Homework Task 2.1**

Have a go at adding some of the following functionality to the project you have been working on today, either in the labs if you have time, or at home before next week's practical session:

1. Create a score variable, initially set to 0, that is incremented when a Pick-up is collected. The score should be displayed for the user to see.
2. When a Pick-up is collected, show it again at a random position on the screen.
3. Allow the spaceship to 'wrap' around the screen i.e. goes out left edge, comes in right and vice versa.
4. Add inertia to the spaceship so it continues to move for a short time after the user releases the forward key. Tip: Try creating a speed variable
5. When the space key is pressed, the spaceship should fire a bullet that moves off in the direction it is currently facing.