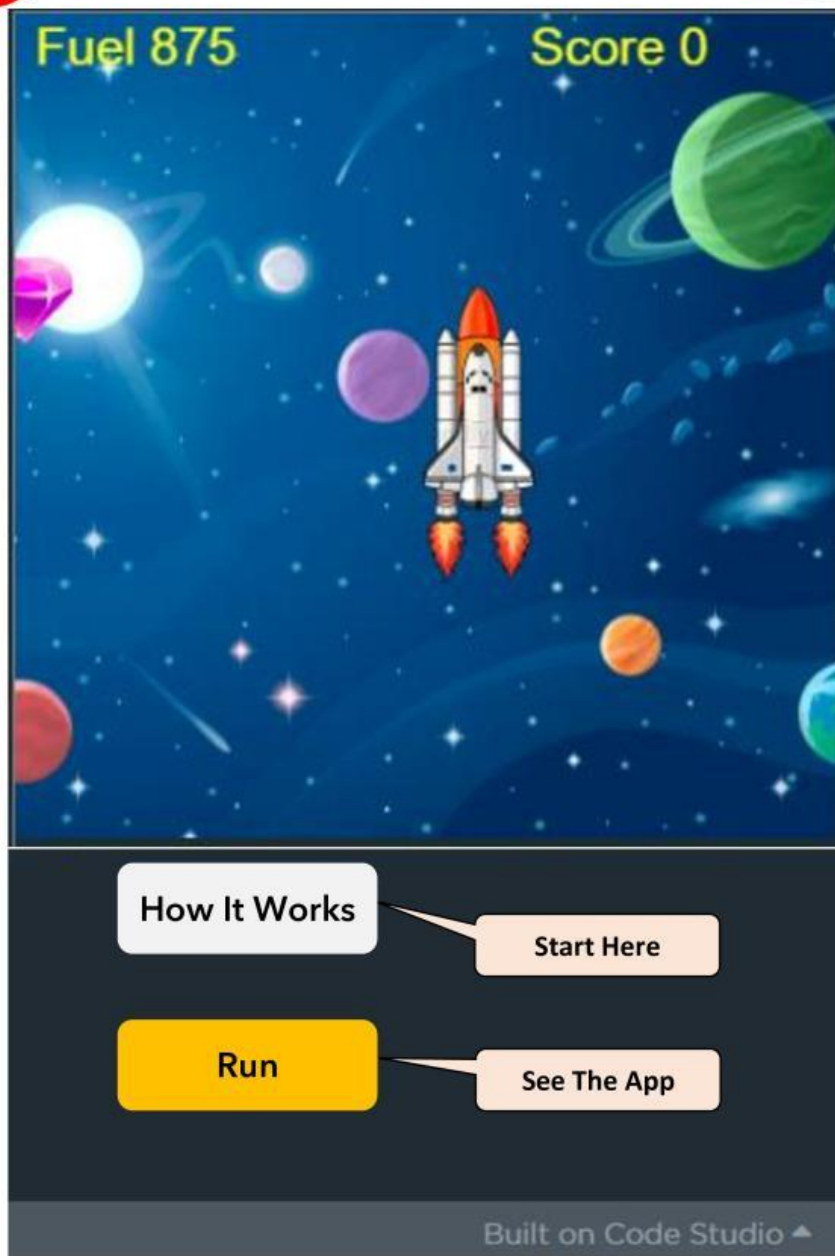


# Project 98

98



## Coding School



- ❖ Let's create the Spaceship Game.
- ❖ Let's create this game to play by collecting gems and collecting points before the spaceship runs out of oil.
- ❖ Let's add a background image first. An image of the space is included in the project given to you in the animation library.

```
var backgroundImage = createSprite(200, 200);
backgroundImage.setAnimation(▼ "Background.jpg_1");
```

- ❖ Then let's create the spaceship sprite.

```
var shuttle = createSprite(200, 340);
shuttle.setAnimation(▼ "Shuttle.png_1");
shuttle.scale = 0.5;
```

- ❖ Let's create the sprite that appears when the game is over.

```
var gameOver = createSprite(200, 200);
gameOver.setAnimation(▼ "game over.png_1");
gameOver.scale = 2;
gameOver.visible = 0;
```

- ❖ Let's define the variables needed to create the app as follows.

```
var countForPurpleGems = 0;
var countForAsteriod = 0;
var countForFuelCan = 0;
var totalPurpleGems = 0;
var totalAsteroids = 0;
var totalFuelCans = 0;
var fuel = 1000;
var countOfPurpleGemOfCollect = 0;
```

- ❖ countFor... variables are created to adjust the time it takes for each item to come from top to bottom, and total... variables are used to store the count of released items.  
The variable called Fuel is equal to 1000 to store the initial oil value and to return it to 1000 when a fuel can is collected again.
- ❖ For each item, define their groups as follows.

```

var gems = createGroup();
var fuelCans = createGroup();
var asteroids = createGroup();

```

- ❖ Create functions related to Gems, Asteroids and fuel can drop as follows.
- ❖ The function related to the design so that the purple gems fall down is as follows.

```

function fallPurpleGems() →
{
  if( countForPurpleGems == 100 )
  {
    countForPurpleGems = 0;
    var purpleGems = createSprite( randomNumber(0,400), 20 ) →;
    purpleGems.setAnimation(▼ "purpleGem.png_1");
    purpleGems.scale = 0.2;
    gems.add(purpleGems);
    gems.setVelocityEach(0, 5);
    totalPurpleGems = totalPurpleGems + 1;
  }
}

```

- ❖ Below is the function related to the design of falling meteorites.

```

function fallAsteroids() →
{
  if( countForAsteriod == 150 )
  {
    countForAsteriod = 0;
    var asteroid = createSprite( randomNumber(0,400), 20 ) →;
    asteroid.setAnimation(▼ "Asteroid.png_1");
    asteroid.scale = 0.4;
    asteroids.add(asteroid);
    asteroids.setVelocityEach(0, 5);
    totalAsteroids = totalAsteroids + 1;
  }
}

```

- ❖ Create the function related to fuel tanks drop as follows.



```

function fallFuelCan() {
  if (countForFuelCan == 300) {
    countForFuelCan = 0;
    var fuelCan = createSprite(randomNumber(0, 400), 20);
    fuelCan.setAnimation(▼ "FuelTank.png_1");
    fuelCan.scale = 0.2;
    fuelCans.add(fuelCan);
    fuelCans.setVelocityEach(0, 5);
    totalFuelCans = totalFuelCans + 1;
  }
}

```

- ❖ Then in the draw function, first let's code to move the spacecraft using the arrow keys.

```

function draw() {
  if (keyDown(▼ "up")) {
    shuttle.setSpeedAndDirection(5, 270);
  }
  if (keyDown(▼ "down")) {
    shuttle.setSpeedAndDirection(5, 90);
  }
  if (keyDown(▼ "left")) {
    shuttle.setSpeedAndDirection(5, 180);
  }
  if (keyDown(▼ "right")) {

```

- ❖ Call the functions created above until the visibility of the game over sprite is not 1.

```

if (gameOver.visible != 1) {
  fallPurpleGems();
  fallAsteroids();
  fallFuelCan();
}

```

This prevents gems, asteroids and fuel cans from falling from above when the game is over.

- ❖ Now let's code what should happen when the spacecraft collides with each object.
- ❖ First, when the shuttle's fuel can is touched, code to increase the amount of fuel to 1000 and destroy the fuel tank.

```

if (fuelCans.isTouching(shuttle)) {
  for (var i = 0; i < totalFuelCans; i++) {
    if (fuelCans.get(i) != undefined && fuelCans.get(i).isTouching(shuttle)) {
      fuelCans.get(i).destroy();
      playSound(▼"sound://category_accent/puzzle_game_accent_b_01.mp3", ▼false); ─
      fuel = 1000;
    }
  }
}

```

- ❖ When the spacecraft touches the gem, code as follows to add the score and to make the related gem disappear and play a sound.

```

if (gems.isTouching(shuttle)) {
  for (var i = 0; i < totalPurpleGems; i++) {
    if (gems.get(i) != undefined && gems.get(i).isTouching(shuttle)) {
      gems.get(i).destroy();
      playSound(▼"sound://category_achievements/lighthearted_bonus_objective_1.mp3", ▼false); ─
      countOfPurpleGemOfCollect = countOfPurpleGemOfCollect + 1;
    }
  }
}

```

- ❖ If the plane collides with a meteor, the plane should be destroyed and game over should be displayed. For that let's code as below.

```

if (asteroids.isTouching(shuttle)) {
  for (var i = 0; i < totalPurpleGems; i++) {
    if (asteroids.get(i) != undefined && asteroids.get(i).isTouching(shuttle)) {
      asteroids.get(i).destroy();
      shuttle.destroy();
      fuel = 0;
      gameOver.visible = 1;
      playSound(▼"sound://category_explosion/8bit_explosion.mp3", ▼false); ─
    }
  }
}

```

- ❖ When the meteorite collides, the meteorite and the plane will be destroyed and the fuel value should be 0. Along with that, the game over sprite should be visible. Create a decaying sound to play.

- ❖ If the fuel level becomes 0, let's prepare the code for the game to be over and the shuttle to crash down.

```

if (fuel == 0)
{
    fuel = 0;
    shuttle.setSpeedAndDirection(1, 90);
    gameOver.visible = 1;
}
else
{
    fuel = fuel - 1;
}

```

- ❖ If the value of the oil becomes 0, then the game is over, otherwise in the else part, the value of the fuel variable is designed to decrease by 1 in each round.
- ❖ Prepare the code as below to display the accumulated points and display the amount of oil.

```

drawSprites();
fill("yellow");
textSize(25);
text("Fuel " + fuel, 10, 25);
text("Score " + countOfPurpleGemOfCollect, 250, 25);

```

- ❖ Finally, increase the value of all the count variables by 1 as follows.

```

countForPurpleGems = countForPurpleGems + 1;
countForAsteriod = countForAsteriod + 1;
countForFuelCan = countForFuelCan + 1;

```