Galaxy Shooter

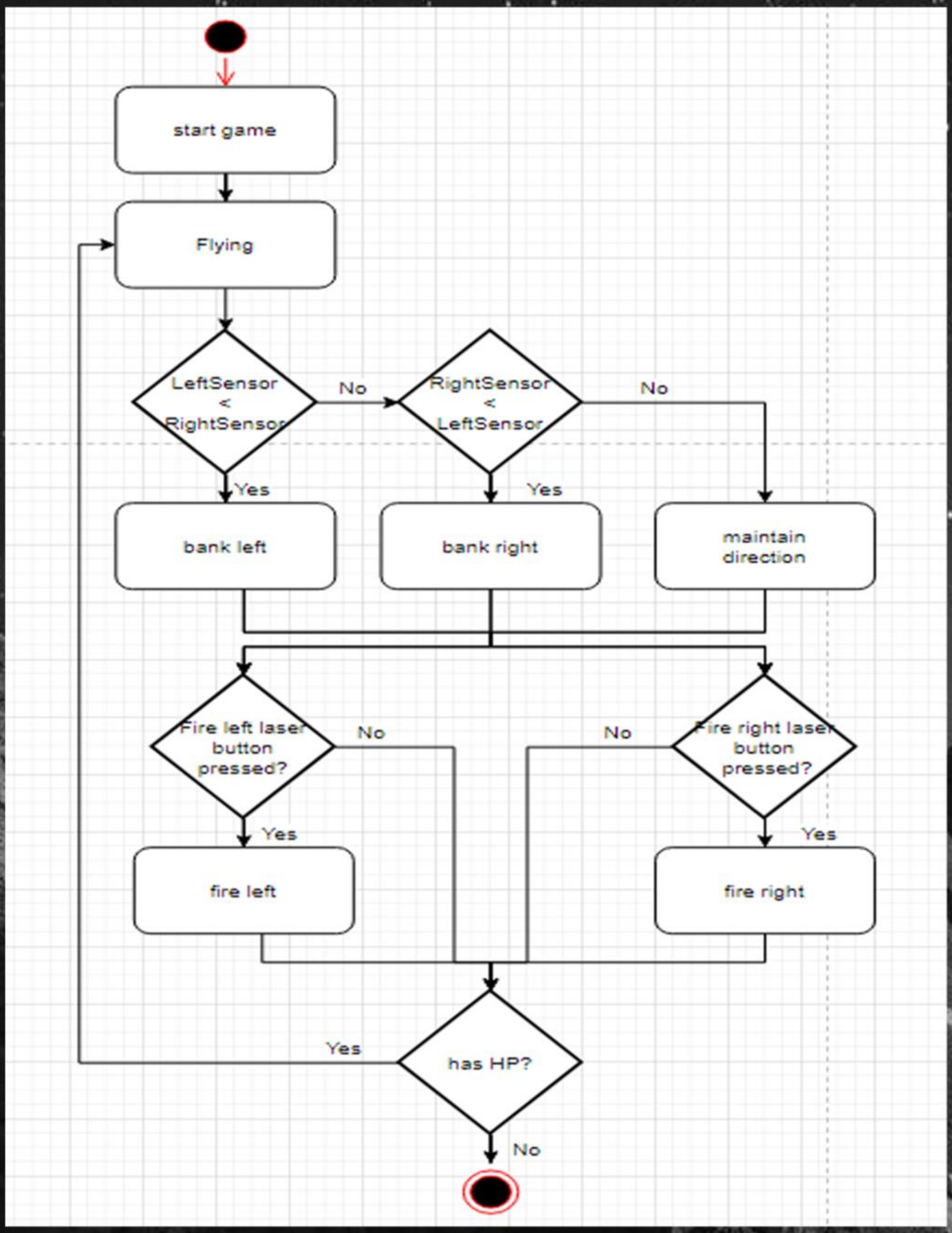
Danny Daley

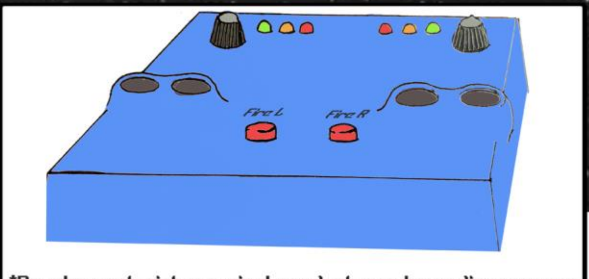
The Game

Galaxy Shooter is an endless runner, navigating through dangerous corners of space occupied by Space Pirates. You will stabilise flight, dodge space debris and incoming projectiles by hovering your hands over the left and right flight stabilisers, while firing from the left and right laser cannon buttons, each with adjustable power output

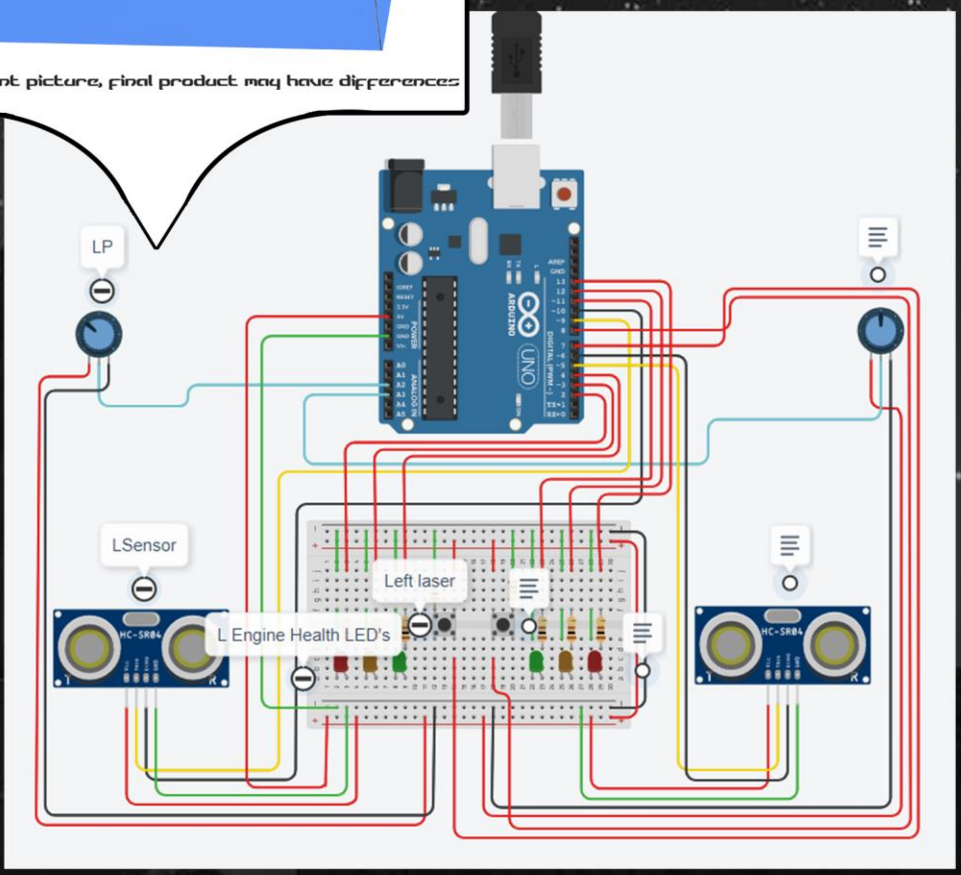
The Controller

The controller for this game utilises 2 Ultrasonic distance sensors that you will use to bank left and right in order to navigate safely through the mission, a button for each the left and right laser cannons with an adjustable power output, controlled by their own potentiometers. Each cannon is also accompanied by a set of green, amber and red LED's to indicate output load on the cannons, overloading could result in the cannons having to cool down before further use, so power them wisely!





*Development picture, final product may have differences



The sensitivity problem

Initially the controller would check which sensor was reading a lower value and then bank accordingly to steer the player ship.
In essence this would be what we would want to happen, but what happens if we want to continue flying straight ahead?
Rather than forcing players to zig-zag toward the incoming enemies, a state machine has been implemented which changes depending on certain sensor values and thresholds. Instead of players having to have their hands completely level over the sensors to allow flight in a straight path forwards, a tolerance threshold was introduced, allowing players to fly forwards even if their hands aren't completely millimetre level.

```
leftSensor:  ←  left sensor reading [0], [400]
rightSensor:  ← right sensor reading [0], [400]
tolerance:  ← 100
lowThreshold:  ← 100
highThreshold:  ← 100

//Flight states
+e FlightStateEnum
{
 BankLeft,
 BankRight,
 IncreaseAltitude,
 DecreaseAltitude,
 Forward
}

//Default Flight state

flightMode ← FlightStateEnum.Forward

//Change states depending on sensor values

IF leftSensor < (rightSensor - tolerance)
 THEN flightMode ← FlightStateEnum.BankLeft

ELSE IF rightSensor < (leftSensor - tolerance)
 THEN flightMode ← FlightStateEnum.BankRight

ELSE IF leftSensor AND rightSensor > highThreshold
 THEN flightMode  ← FlightStateEnum.DecreaseAltitude

ELSE IF leftSensor AND rightSensor < lowThreshold
 THEN flightMode ← FlightStateEnum.IncreaseAltitude
ELSE

 THEN flightMode  ← FlightStateEnum.Forward
```