

# Instructions



Build and code ships to fight in battle against other players' ships.

## Getting Started

To start, register a new account from the **Sign In** page.

After that's done, you're going to want to create your first ship. To do this, go to the **Ship Editor**. In the ship editor, you can build and code your ship. Keep an eye on the **budget** in the top left, and check out these [example ships](#) to get an idea of how the Lua scripting works. The '**TEST**' button on the bottom right allows you to test the currently loaded ship in battle against any of your other ships, without affecting your multiplayer rating. Testing a ship by itself isn't very useful, so try creating a square out of hull blocks for target practice until you've built more than a single ship.

Once you're done building your ship and have verified that it works in testing, you're ready to send it off to battle. Press *Escape* and exit to the main menu, then click '**To Battle**'. You're presented with a dialog where you're able to choose three ships to fight with. You must select three ships, and you may select the same ship multiple times. Once you have your ship(s) selected, hit '**Fight!**' and the game will enter battle with your selected ships. Your ship is now set to public, and anyone can fight against it. Multiplayer is not real-time, and your **rating** can change while you're not playing from other people fighting against your ships. Feel free to make changes to your ships at any time from the Ship Editor.

## Editing Scripts With an External Editor

We understand that the built-in editor on the **SCRIPT** tab isn't all that great, and fully support the use of external text editors like Sublime Text and Notepad++ to edit your ships' script files. You can find the script files on Windows in the **Mast\_Data/Ships/** directory. Edit the .lua files directly while tabbed out of the game, and when you tab back in your changes will be updated in-game. Ignore the .json files.

# Controls

## ● Any Screen

- Escape: *Open escape menu*

## ● Ship Editor

- Right Click + Drag, Middle Click + Drag, WASD: *Pan camera*
- Mouse Wheel, +/-: *Zoom camera*
- Left Click: *Select part / place part*
- Shift (*hold*) + Left Click: *Multi-place parts*
- R: *Rotate part*
- Left Control (*hold*): *Weld/unweld parts*
- Delete, Backspace: *Delete part / cancel placement*
- Right Click on a part: *Set / Change part name*

## ● Battle

- Right Click + Drag, Middle Click + Drag, WASD: *Pan camera*
- Mouse Wheel, +/-: *Zoom camera*
- T: *Reset camera*
- F: *Toggle automatic camera*
- 1-6: *Follow a ship*
- Q (*hold*): *Slow motion*
- E (*hold*): *Fast forward*

# Scripting API

## Parts

### ● Global (Common to every part)

#### ○ bool .alive

■ A boolean indicating whether the part is alive (true) or dead (false).

### ● Brain

#### ○ void :SetColor(int positionX, int positionY, Color color)

■ Sets the color of the brain indicator at this position (upper left is 0,0) to the color.

#### ○ Color :GetColor(int positionX, int positionY, Color color)

■ Sets the color of the brain indicator at this position (upper left is 0,0) to the color.

#### ○ void :SelfDestruct()

■ Destroys the brain.

#### ○ bool :GetRoundOver()

■ Returns true if the round is over.

#### ○ bool :GetVictory()

■ Returns true if this ship has won the round.

### ● Thruster

#### ○ void :SetThrust(float ratio)

■ Sets the thrust ratio for this thruster, with 0 being no thrust and 1 being full thrust.

#### ○ float :GetThrust()

■ Gets the current thrust ratio for this thruster.

### ● Big Thruster

#### ○ void :Ignite()

■ Ignites the thruster. Once ignited, it goes at full force until it runs out of charge 3 seconds later, at which point it has to recharge for 5 seconds before it can be ignited again.

#### ○ float :GetCharge()

■ Get the current charge level of the thruster (0-1).

● **Turret / Big Turret / Rocket Launcher**

○ void :Fire()

■ *Fire a single projectile.*

● **Ranger**

○ float :Range()

■ *Returns the current distance to the hit object.  
Returns -1 if nothing was hit.*

○ bool :IsAlly()

■ *Returns true if the hit object is an allied part,  
otherwise returns false.*

● **GPS**

○ Vector2 :Pos()

■ *Returns the position of the GPS as a Vector2.*

○ Vector2 :Vel()

■ *Returns the velocity of the GPS as a Vector2.*

● **Gyroscope**

○ float :Ang()

■ *Returns the rotation of the gyroscope (in degrees) as  
a float.*

○ float :AngVel()

■ *Returns the angular velocity (in degrees per second)  
of the gyroscope as a float.*

● **Decoupler**

○ void :Decouple()

■ *Detach all parts welded to the decoupler. Note that  
this applies a slight outward force.*

● **Heal Beam**

○ void :SetEnabled(bool enabled)

■ *Turn the healing beam on (true) or off (false).*

# Event Functions

- `function Start()`
  - This function is called once at the start of execution.
- `function Update()`
  - This function is called periodically, many times per second.

# Functions

- `void print(string message)`
  - *Print a message to the debugging panel on the bottom of the screen. Unity's rich text styling is supported.*
- `float time()`
  - *Returns the time in seconds since the start of battle.*

# Contact

Questions? Comments? Concerns?

You can reach us at [propslam@gmail.com](mailto:propslam@gmail.com).