



Modular Robot Builder & Combat Kit

Documentation · version 1.0.0 · Unity 6 URP · Hady Studios · hadystudios.com



1. Quick start

1. Use a Unity 6 project with the **Universal Render Pipeline** (the "Universal 3D" template is perfect).
2. Import the package. The kit needs **Input System**, **Cinemachine** 3 and uGUI: the Unity Asset Store installs them with the kit. (From Fab, itch.io or a .unitypackage by hand, the Setup window opens and installs them with one click; until then only the Setup window compiles, so there are no errors.) If Unity asks to enable the new Input System backends, click **Yes** (the Editor restarts).
3. The **Setup** window opens by itself (later: *Tools > Robot Combat Kit > Setup*). It checks your project and offers to add the kit's scenes to the Build Settings.
4. Open `Assets/RobotCombatKit/Scenes/MainMenu` and press **Play**. That's the whole game: garage, career, fights.

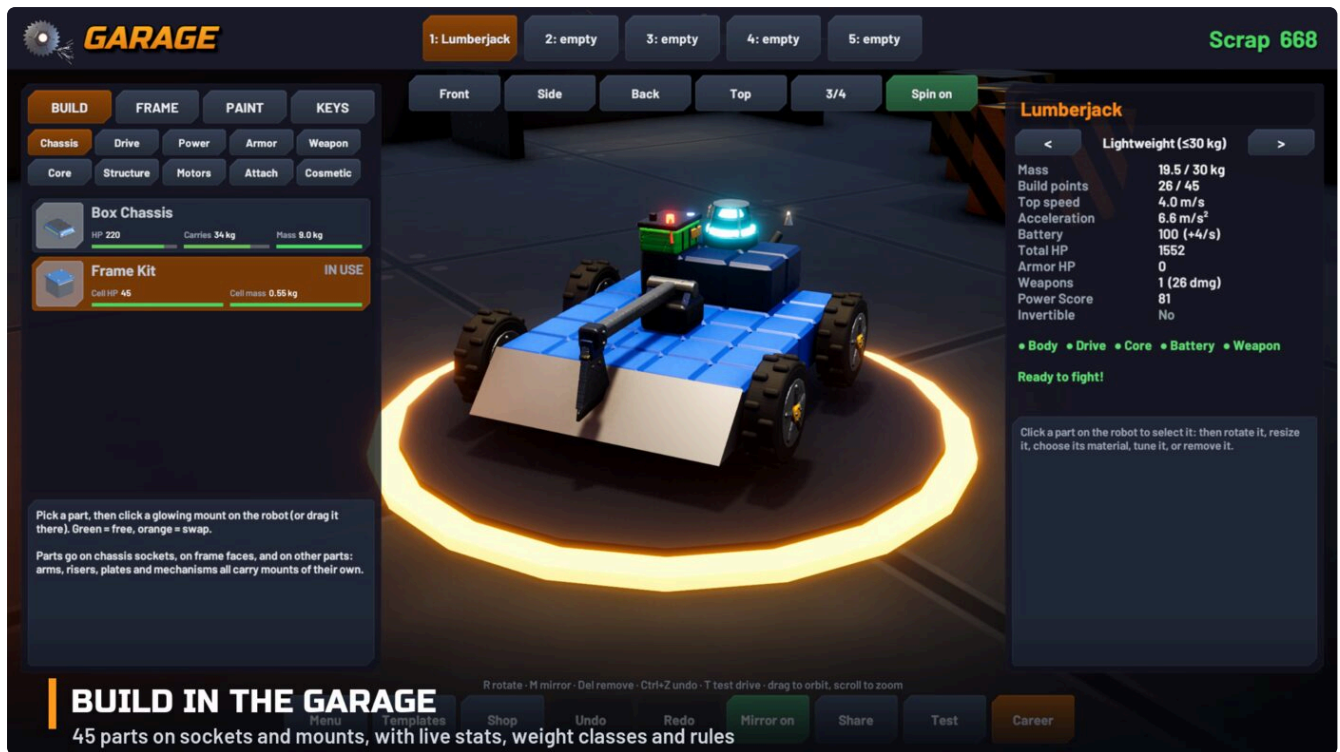
You can press Play in any kit scene (Garage, Arena_01, the demos). The kit's data loads itself, and in the Editor the scenes load even before they are in the Build Settings. For a build, add them with the Setup window first.

The demo scenes

Scene	What it shows
MainMenu	The full game: menu, garage, career, shop, settings.
Garage	The robot builder on its own.
Arena_01	A fight with the current robot against an opponent from the career.
TestArena	Test drive against a harmless dummy.
Demo_Showroom	Every part on a plinth, by category. Tab or the arrow keys jump between rows.
Demo_QuickFight	Two AIs fight straight away. Press F2 to take over the blue robot. Choose the opponent, or paste a blueprint code for your robot, on the <code>QuickFight</code> object.

Controls

Action	Keyboard	Gamepad	Touch
Drive (tank steering)	W A S D / arrows	Left stick	Thumb stick (anywhere on the left)
Weapons 1 · 2 · 3	Space · Q · E (and F, G, X)	A / RT · X / LT · B	Weapon buttons
Self-right	R	Y	Button
Camera (arena / close)	C	Select	CAM
Pause	Esc	Start	Pause button
Spectator (AI drives you)	F2		
Garage: rotate, mirror, remove, undo	R · M · Del · Ctrl+Z		Buttons on the part



2. How it fits together

Everything lives under `Assets/RobotCombatKit/`, namespace `RobotCombatKit`. The code is in one assembly (`RobotCombatKit.asmdef`) with an Editor assembly and a test assembly beside it.

Folder	What's inside
<code>Scripts/Core</code>	<code>GameManager</code> (survives scene loads, owns the save and the config), <code>GameConfig</code> , save system, settings, game events, procedural audio, VFX, music.
<code>Scripts/Parts</code>	<code>PartDefinition</code> and its subclasses (one per category), <code>PartDatabase</code> , mounts and sockets, part health, dents, paint.
<code>Scripts/Robot</code>	<code>RobotDesign</code> (plain data), <code>RobotAssembler</code> (data → working robot), <code>RobotController</code> , <code>RobotStats</code> , frame-kit bodies, inputs.
<code>Scripts/Movement</code>	Wheels and treads (<code>WheelColliders</code> upright, a raycast model upside down).
<code>Scripts/Weapons</code>	Flipper, spinner, hammer, saw, ram, and the mechanisms (hinges, spin motor, piston, turntable).
<code>Scripts/AI</code>	<code>AIProfile</code> (an opponent: how it drives and the robot it brings), <code>AIRobotInput</code> .
<code>Scripts/Arena</code>	<code>MatchManager</code> (rules), <code>ArenaBuilder</code> , hazards, judges, camera, crowd, scoreboard.
<code>Scripts/Builder</code>	The garage: <code>GarageController</code> , orbit camera, socket markers.
<code>Scripts/Inventory</code>	Inventory, shop, rewards, career, challenges, templates, blueprint share codes.
<code>Scripts/UI</code>	Every screen, built from code with <code>UIKit</code> . Touch controls.

Scripts/Demo	The showroom and quick-fight demos.
Data	The ScriptableObjects: parts, AI opponents, robot physics, the part database.
Resources/RobotCombatKit	GameConfig (ties all the data together), fonts, logo.
Models · Materials · Textures · UI/Icons	The art. Models import through Editor/ModelImportRules.

The flow of one robot

```

RobotDesign (data: part ids, mounts, paint) —RobotAssembler.Build→ RobotController
(one Rigidbody)
    ▲ saved as JSON, shared as an RCK1: code
    PartHealth + colliders
    |
    (WheelDrive, TreadDrive)
    Garage edits it
    (WeaponPart.Tick every physics step)

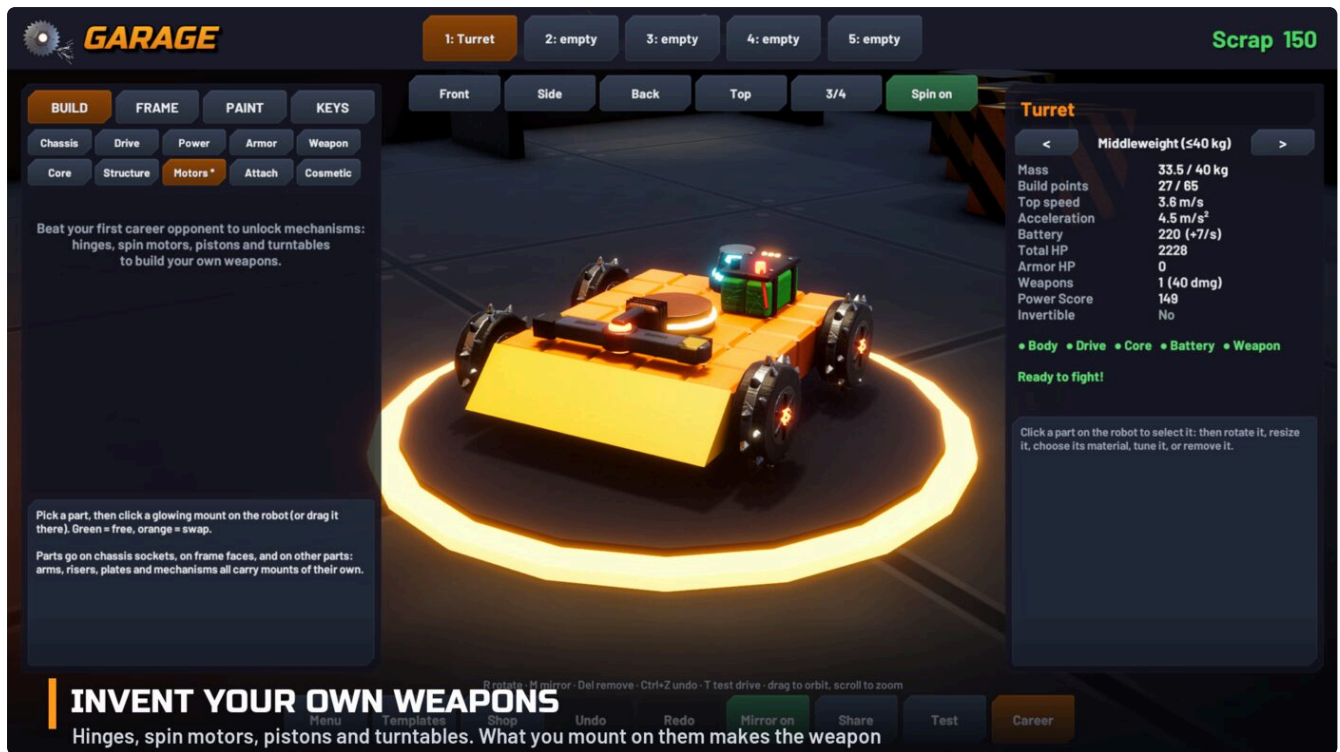
PlayerRobotInput or AIRobotInput
    |
    └─ parts: RobotPart +
    └─ drive parts
    └─ weapons
    └─ IRobotInput:

```

A robot never stores asset references: designs hold part **ids**, looked up in the `PartDatabase`. So saves, share codes and (later) network messages stay plain data. The controller doesn't know who drives it. The player and the AI are both just an `IRobotInput`.

Damage

Every part has its own `PartHealth`. A collision is routed to the part that was actually hit, and how hard it was depends on the relative speed, the mass, the weapon and the armour. `PartDents` presses dents into the mesh where a hit lands (the mesh is copied on the first hit, so untouched parts cost nothing). A broken part detaches and everything mounted on it falls off with it. When the Core is destroyed, the robot loses.



Match rules

MatchManager wins a fight on: the enemy Core destroyed, the enemy in the pit or out of the arena, the enemy immobile for 10 s, or, when time runs out, the judges (JudgesScore : damage, aggression, control). Change the lengths and limits on the MatchManager in the arena scene.

3. Common jobs

Change a part's numbers

Select its asset in Data/Parts and edit it in the Inspector: mass, health, price, build points, damage, speed, and so on. Each field has a tooltip.

Add a new part

1. Model it with its root at the mounting point, front facing +Z, 1 unit = 1 m. Name the materials like the kit's (M_Steel, M_Paint ...) so the import rules swap in the kit's URP materials. Collision shapes are meshes named COL_... . Mounts on the part are empties named Node_Out_0 / Node_Deck_0 , and moving pieces are child objects with the pivot as their origin.
2. Drop the FBX into Models/ . The import rules swap in the materials, hide the COL_ shapes and turn them into colliders.
3. Create › Robot Combat Kit › Parts › ... : pick the category, give it a unique id, a name, an icon and the model (the FBX, or a prefab of it, in the Prefab field), and set its numbers.
4. Add it to Data/PartDatabase (the parts list). It now shows in the garage and the shop, and in the Showroom.

Tip: the quickest new weapons need no code at all. Mount attachments on the existing mechanisms. A new *kind* of weapon is a subclass of `WeaponPart` (override `Tick`), set up in `RobotAssembler` where the other weapons are.

Add an opponent

Create › *Robot Combat Kit* › *AI Profile*. Set its id, name, colour, weight class, difficulty, reaction time, aim and aggression. For its robot, build it in the garage, press **Share** to copy its code, paste the code into the profile's **Blueprint Code** field and choose **Use Blueprint Code** from the asset's context menu. Add the profile to `GameConfig`'s `career` list, and to an event in one of its `leagues`. Rivals (**Is Rival**) get an introduction with their own lines before the fight.

Change the career

`GameConfig` holds the `leagues`. Each league has events (duel, short fuse, hazard night, gauntlet), each with its opponent ids (in order, for a gauntlet), story text and scrap reward. Winning a league's last event wins its trophy, with trophy scrap and a prize part. Each league has a weight class. `GameConfig` also sets the starting scrap, starter parts, the starter robot and the templates.

Make a new arena

Duplicate `Arena_01`. Its `ArenaBuilder` builds the floor, walls, pit and hazards from a few numbers: change them and use the component's context menu **Build**. Move the spawn points as you like. To use it in the game, load your scene name instead of `SceneNames.Arena`.

Change the look of the UI

All screens are uGUI made from code. The colours are in `Theme` (in `UIKit.cs`) and the fonts and logo are in `Resources/RobotCombatKit`. Swap `Brand/Logo.png` and `Brand/Emblem.png` for your game's own.



Mobile

On phones the kit shows touch controls, uses the safe area, runs at 60 fps and keeps the screen on. Recommended player settings: landscape only, IL2CPP, ARM64. On Android, a *Forward* URP renderer is the safest choice for older GPUs, with up to 8 additional lights per object (the arena floor is lit by nine floodlights).

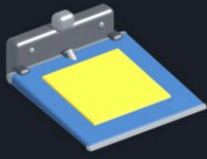
4. Good to know

- **Physics.** Small robots need a fine physics step: at start-up the kit sets `Time.fixedDeltaTime` to 0.01 and raises the solver iterations (`GameManager.ApplyPhysics`). Change it there if your game needs something else.
- **macOS.** The kit turns the SRP Batchers off at runtime on macOS (Metal drew the arena floor black on some loads with it on). Other platforms keep your setting.
- **Save file.** `robotkit_save.json` in `Application.persistentDataPath`.
- **Keep the folder where it is.** The model import rules look for `Assets/RobotCombatKit/Models`.
- **Tests.** `Window > General > Test Runner > EditMode`: 23 tests check the data (every part has a model and an icon, every career robot is valid, share codes round-trip...).
- **Recommended render settings** (Setup window): shadow distance 25 m, 2 cascades, HDR, 8 lights per object.

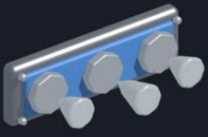
5. All parts

WEAPONS AND MECHANISMS

10 parts · Modular Robot Builder & Combat Kit



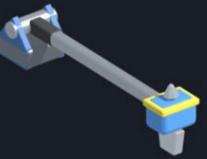
WEAPON
Flipper
3.5 kg · 80 HP · 80 scrap



WEAPON
Ram Spike
2 kg · 140 HP · 40 scrap



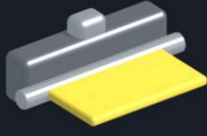
WEAPON
Circular Saw
3 kg · 70 HP · 120 scrap



WEAPON
Hammer
4 kg · 90 HP · 140 scrap



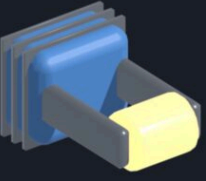
WEAPON
Disc Spinner
5.5 kg · 110 HP · 300 scrap



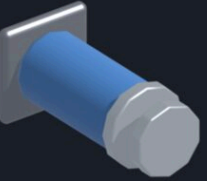
MECHANISM
Lift Hinge
2.2 kg · 80 HP · 90 scrap



MECHANISM
Swing Hinge
2.4 kg · 80 HP · 100 scrap



MECHANISM
Spin Motor
2.6 kg · 90 HP · 160 scrap



MECHANISM
Piston
1.8 kg · 90 HP · 80 scrap



MECHANISM
Turntable
1.6 kg · 90 HP · 140 scrap

Chassis (4)

Part	Id	Mass	HP	Build pts	Price	What it does
Box Chassis	chassis_box	9.0 kg	220	6	120	A sturdy all-rounder. Front, rear and top weapon mounts.
Wedge Chassis	chassis_wedge	10.0 kg	200	6	220	Low sloped nose that slides under opponents. Angled armor deflects some damage.
Low Rider	chassis_lowrider	12.0 kg	260	6	380	Wide, flat and heavy. Hard to flip and carries the most weight.
Frame Kit	frame_kit	0.0 kg	0	0	0	Build the body yourself from 10 cm cells: cubes, wedges and slabs in five materials.

Movement (3)

Part	Id	Mass	HP	Build pts	Price	What it does
Rubber Wheel	wheel_rubber	1.0 kg	45	2	30	A grippy rubber wheel. Cheap, light and quick.
Spiked Wheel	wheel_spiked	1.4 kg	70	2	90	Steel spikes bite into the floor: more grip and pushing power, a bit slower.
Tread	tread_standard	3.5 kg	120	2	150	One side of a tank tread. Fit a pair. Huge grip and very tough, but slow.

Power (2)

Part	Id	Mass	HP	Build pts	Price	What it does
Small Battery	battery_small	1.5 kg	40	2	40	Enough energy for a short fight.
Large Battery	battery_large	3.2 kg	70	2	130	Twice the capacity and faster recharge, at twice the weight.

Armor (3)

Part	Id	Mass	HP	Build pts	Price	What it does
Steel Plate	armor_steel	3.0 kg	90	1	50	Heavy steel. Blocks a third of every hit.
Titanium Plate	armor_titanium	2.0 kg	110	1	160	Lighter and tougher than steel. Blocks half of every hit.
Wedge Plate	armor_wedge	3.5 kg	100	1	90	A ramp that lifts attackers off their wheels.

Weapon (10)

Part	Id	Mass	HP	Build pts	Price	What it does
Flipper	weapon_flipper	3.5 kg	80	4	80	Pneumatic arm that throws opponents into the air. Also lets you self-right.
Ram Spike	weapon_ram	2.0 kg	140	2	40	No moving parts. The faster you hit, the more it hurts.
Circular Saw	weapon_saw	3.0 kg	70	4	120	Hold fire to grind through armor. Low damage per touch, but it never stops.
Hammer	weapon_hammer	4.0 kg	90	4	140	Top-mounted. Swings down on whatever is in front of you. Great against cores.
Disc Spinner	weapon_spinner	5.5 kg	110	5	300	A heavy disc that stores spin energy and launches what it hits. Takes 3 s to spin up.
Lift Hinge	act_lift_hinge	2.2 kg	80	3	90	Rests flat, snaps up. Put a plate on it for a flipper, a wedge for a lifter. Can self-right.
Swing Hinge	act_swing_hinge	2.4 kg	80	3	100	Stands on a deck and slams down. Put a hammer head or an axe on the end.
Spin Motor	act_spin_motor	2.6 kg	90	3	160	Spins whatever you mount on it. Heavier and wider = harder hits, slower spin-up. Turn it to spin flat.
Piston	act_piston	1.8 kg	90	3	80	Punches its rod straight out. Put a spike on the end for a rammer.
Turntable	act_turntable	1.6 kg	90	3	140	Turns its top to face the opponent: anything on it aims itself. Key toggles tracking.

Core (1)

Part	Id	Mass	HP	Build pts	Price	What it does
Control Core	core_control	1.2 kg	100	1	60	The robot's brain. Exactly one per robot. If it's destroyed, you lose.

Structure (4)

Part	Id	Mass	HP	Build pts	Price	What it does
Riser	struct_riser	0.5 kg	60	1	20	Raises whatever stands on it. Mounts on all four sides and on top.
Extension Arm	struct_arm	0.7 kg	70	1	25	Puts a weapon or armor further out. Resizable in length.
Angle Mount	struct_angle	0.4 kg	50	1	20	Tilts what's on it 30° down. Turn it over to tilt up.
Mounting Plate	struct_plate	0.9 kg	80	1	30	A flat plate with four deck mounts and four edge mounts.

Attachment (8)

Part	Id	Mass	HP	Build pts	Price	What it does
Flipper Plate	att_plate	0.9 kg	70	1	30	A flat plate that reaches forward. On a lift hinge it's a flipper.
Scoop Wedge	att_wedge	1.0 kg	90	1	35	A low ramp that slides under opponents.
Spinner Disc	att_disc	2.0 kg	110	1	90	A heavy disc for a spin motor.
Spinner Bar	att_bar	1.4 kg	90	1	80	A long bar for a spin motor: fast, deadly to wheels when spun flat.
Spinner Drum	att_drum	2.4 kg	130	1	90	A wide drum for a spin motor: low and stable.
Hammer Head	att_hammer_head	1.4 kg	100	1	40	A heavy head for a swing hinge.
Axe Blade	att_axe	0.8 kg	70	1	60	Less mass than a hammer, much more bite.
Spike	att_spike	0.5 kg	90	1	25	On a piston it's a rammer. Bolted straight on, it hurts whatever you drive into.

Cosmetic (10)

Part	Id	Mass	HP	Build pts	Price	What it does
------	----	------	----	-----------	-------	--------------

Antenna	<code>cos_antenna</code>	0.1 kg	15	0	15	Looks only. Never changes your stats.
Horns	<code>cos_horns</code>	0.1 kg	15	0	15	Looks only. Never changes your stats.
Glowing Eyes	<code>cos_eyes</code>	0.1 kg	15	0	15	Looks only. Never changes your stats.
Spoiler	<code>cos_spoiler</code>	0.1 kg	15	0	30	Looks only. Never changes your stats.
Team Flag	<code>cos_flag</code>	0.1 kg	15	0	15	Looks only. Never changes your stats.
Number Plate	<code>cos_number</code>	0.1 kg	15	0	15	Looks only. Never changes your stats.
Spike Crest	<code>cos_mohawk</code>	0.1 kg	15	0	30	Looks only. Never changes your stats.
Exhaust Pipes	<code>cos_exhaust</code>	0.1 kg	15	0	30	Looks only. Never changes your stats.
Siren Light	<code>cos_siren</code>	0.1 kg	15	0	45	Looks only. Never changes your stats.
Sticker	<code>cos_sticker</code>	0.0 kg	5	0	10	A decal for any face. The Design slider picks star, bolt, skull, target, teeth or flame; it takes your accent colour. Resize it as big as you like.

6. Credits and support

Made by Hady Studios. Everything in the kit is original: models from our own Blender scripts, textures generated by code, sound synthesised at runtime. The only third-party files are the UI fonts **Barlow** and **Russo One**, free under the SIL Open Font License 1.1 (licence texts in `Resources/RobotCombatKit/Fonts`, see `Third-Party Notices.txt`).

Questions or problems: write to **support@hadystudios.com** or use the form at **hadystudios.com/support**. We reply within 48 hours, and fixes go into free updates. On **hadystudios.com** you can also play the kit in your browser, read this guide and see our other packs.

The kit is licensed to you under the licence of the store you got it from: the Unity Asset Store EULA, the Fab End User License Agreement, or the licence on its itch.io page. In short: use it in as many games as you like, commercial or not; don't resell or share the source files themselves.