PLATFORM CRAFTER

USER GUIDE

Unity's Platform Modular System

What is the Platform Crafter Modular System?

The Platform Crafter Modular System is a package for Unity 2D with the goal of creating a universal tool capable of recreating old and new 2D platformers of the gaming world or help create a platformer game from scratch. The system is supposed to be intuitive and easy to use, while keeping the necessary complexity for creating a controller capable of enacting mechanics and functionalities.

It's readily available and meant for all users of Unity, be it a knowledgeable user or a new one. If you wish to use its base functionalities, you can do so, or if you want to expand it and use it in conformity with your code.

Modular Brain

The Modular Brain is the core of the operation. Once an object has a brain, it is easy to start adding what are called **Modules**. These modules act as operations that dictate what the entity can do. The Brain shows you how many **Modules** it has and displays them categorically. Each Module can be activated or deactivated, and act as plug-and-play, where it is easy to switch between settings of each module.

The modules are Scriptable Objects; thus, they retain data and can be created, with each one having its own setting and usability.

Getting Started

- Drag the **Modular Brain** script onto the desired game object, or search for it on the **Add Component**. Note: The Modular Brain will add all the necessary components to the object if it doesn't have them.

- Before adding modules, it's important to know what the Modular Brain has. First, right on the top, the number of **Modules** currently on the Brain, divided by their **Type**, with an icon and a number below. Then, the **Folders**, which indicate the **Categories** of the modules, with the categories being **Physics**, **Action/Interaction**, **Container**, **Visuals & Sounds** and **Custom** Modules.

- We can start by opening the **Physics Category Folder**. This category has two **Type Modules**, the **Horizontal Movement** and the **Vertical Movement**. The Brain can only have one of each of these types of modules at a time. Adding it shows the settings of the module.

- Add a **Horizontal Movement**. You can do this by clicking **Right-Mouse** on the Folders window of Unity, and clicking the “Platformer Crafter’s Modular System > Modules > Type – HM” tab. This will create a Scriptable Object on the directory that is opened. Rename this object.

- The Module can be altered outside the Brain, or you can **drag** the object onto the empty slot named Horizontal Movement or find it through the filtered search.

- Once the module is installed, it can be replaced by another or removed completely. The settings of the modules will then be displayed on a fold out and can be changed. Note: the settings will only show on the brain if it’s installed.

- On the top-right corner of the module display, there is a square button, this button activates and deactivates the module. The white light means active, and greyed-out means inactive.

- The process of adding each type of Modules is the same, just look for the type that it represents.

- If you wish to, you can Toggle Editor Features on the Script Menu of the Modular Brain script by clicking on the 3 dots in the corner.

Physics Category Modules

Horizontal Movement Type Module

The HM Module serves as the fundamental module for walking and running of an entity. This Type Module has actions that dictate what the entity does. The Module has a default action named **Walk**, used for walking by pressing the left or right keys. The module uses the Rigid Body 2D component and a 2D Collider, namely Box or Capsule. You can only have **one** of these Type Modules per Modular Brain.

General Settings:

- **Left Key:** Selects the key for moving on the left direction relative to the X axis.
- **Right Key:** Selects the key for moving on the right direction relative to the X axis.
- **Extra Actions:** Selects the extra actions to be active on the module.
- **Ground Layer:** Layer that allows the character to move.
- **Ground Check Range:** Range between the surface and the entity.
- **Can Move On Air:** By enabling this, the entity can be moved horizontally by input whilst not touching the ground.

Walk Action

The Walk Action of the HM Module allows for the basic horizontal movement of the entity. This changes the HM State to Walking. Walk Action has three distinct modes that it can operate with:

Constant Speed: A constant movement in a certain direction.

Constant Speed Settings:

- **Speed:** Base speed of the constant movement.

Acceleration Speed: Movement that gets progressively faster by moving and slower by not pressing any input.

Acceleration Speed Settings:

- **Speed:** Base speed of the accelerating movement.
- **Max Speed:** Amount to which the velocity gets capped and doesn't go above.
- **Acceleration:** How much velocity the entity gets by moving over time.
- **Deceleration:** How much velocity the entity loses over time.

Vehicle-Like: Like the Acceleration Speed, but there is an extra input named Brake that allows for braking.

Vehicle-Like Settings:

- **Brake Input:** Select a key input for braking.
- **Horizontal Brake:** By enabling this, you can brake using the opposite horizontal key input of the current moving one.
- **Speed:** Base speed of the Vehicle-Like movement.
- **Max Speed:** Amount to which the velocity gets capped and doesn't go above.

- **Acceleration:** How much velocity the entity gets by moving over time.
- **Deceleration:** How much velocity the entity loses over time.
- **Brake Force:** How much force is used to stop the movement of the entity on brake input.

Sprint Action

The Sprint Action of the HM Module allows for the mechanic of running by input. This changes the HM State to Sprinting.

General Settings:

- **Sprint Key:** Selects a key input for the action of Sprint.
- **Allow Double Tap:** By enabling this, the entity can go into the state of Sprinting by clicking one of the horizontal inputs twice on a small-time interval.

Sprint Action has two distinct modes that it can operate with:

Constant Speed: A constant movement in a certain direction.

Constant Speed Settings:

- **Speed:** Base speed of the constant movement.

Acceleration Speed: Movement that gets progressively faster by moving and slower by not pressing any input.

Acceleration Speed Settings:

- **Speed:** Base speed of the accelerating movement.

- **Max Speed:** Amount to which the velocity gets capped and doesn't go above.
- **Acceleration:** How much velocity the entity gets by moving over time.
- **Deceleration:** How much velocity the entity loses over time.

Dash Action

The Dash Action of the HM Module allows for the entity to perform a dash on a one of the horizontal directions by input. This changes the HM State to Dashing.

General Settings:

- **Dash Key:** Selects a key input for dashing.
- **Allow Double Tap:** By enabling this, the entity can go into the state of Dashing by clicking one of the horizontal inputs twice on a small-time interval.
- **Shadow Effect:** By enabling this, the entity will use a special effect named Shadow Effect, a separate component of the entity that creates a set of after images when dashing. Check the page _ for more details about this component.
- **Dash Distance:** How far the entity will go horizontally when dashing.
- **Dash Speed:** How fast the entity performs the dash.
- **Cooldown:** Time between each dash to be able to be performed.

Note: the dash can be performed on the air, leading to an **Air Dash**.

Vertical Movement Type Module

The Vertical Movement module serves as the fundamental module for jumping and crouching. This Type Module has actions that dictate what the entity does. The Module has a default action named **Jump**, used for jumping by pressing the jump key. The module uses the Rigid Body 2D component and a 2D Collider, namely Box or Capsule. You can only have **one** of these Type Modules per Modular Brain.

General Settings:

- **Jump Key:** Selects a key input for the action of Jump.
- **Extra Actions:** Selects the extra actions to be active on the module.
- **Ground Layer:** Layer that allows the character to move.
- **Ground Check Range:** Range between the surface and the entity.

Jump Action

The Jump Action of the VM Module allows for the entity to perform a single jump by input while on the ground. This changes the VM State to Jumping. Jump Action has two distinct modes that it can operate with:

Constant Height Jump: A jump that always hits the same height no matter the input.

Constant Height Jump Settings:

- **Jump Height:** The height to which the jump can reach. Note that this value can change depending on the gravity scale.
- **Gravity Scale:** Rigid body's entity gravity scale when on the state of jumping up (+y).
- **Fall Gravity Scale:** Rigid body's entity gravity scale when on the state of falling down (-y).

Derivative Height Jump: A jump that can be derive the height depending on the input press time.

Derivative Height Jump Settings:

- **Initial Jump Force:** Applied force on the start of the jump.
- **Gravity Scale:** Rigid body's entity gravity scale when on the state of jumping up (+y).
- **Fall Gravity Scale:** Rigid body's entity gravity scale when on the state of falling down (-y).
- **Max Jump Duration:** For how long the input can be maintained and the jump performed. Increasing this might lead to an almost "flight" mechanic.

Air Jump Action

The Air Jump Action of the VM Module allows for the entity to perform a jump by input while off ground. This changes the VM State to Air Jumping.

General Settings:

- **Air Jump Key:** Selects a key input for the action of Jump.
- **Max Extra Jumps:** How many air jumps can be performed in total before having to touch the ground to reset.

Air Jump Action has two distinct modes that it can operate with:

Constant Height Jump: A jump that always hits the same height no matter the input.

Constant Height Jump Settings:

- **Jump Height:** The height to which the jump can reach. Note that this value can change depending on the gravity scale.
- **Gravity Scale:** Rigid body's entity gravity scale when on the state of jumping up (+y).
- **Fall Gravity Scale:** Rigid body's entity gravity scale when on the state of falling down (-y).

Derivative Height Jump: A jump that can be derive the height depending on the input press time.

Derivative Height Jump Settings:

- **Initial Jump Force:** Applied force on the start of the jump.
- **Gravity Scale:** Rigid body's entity gravity scale when on the state of jumping up (+y).
- **Fall Gravity Scale:** Rigid body's entity gravity scale when on the state of falling down (-y).

- **Max Jump Duration:** For how long the input can be maintained and the jump performed. Increasing this might lead to an almost “flight” mechanic.

Crouch Action

The Crouch Action of the VM Module allows for the entity to crouch down, limiting its movement and collider height. This changes the VM State to Crouching.

General Settings:

- **Crouch Key:** Selects a key input for the action of Crouch.

Crouch Action has two distinct modes that it can operate with:

Normal Crouch: A simple crouch action that reduces height and movement.

Normal Crouch Settings:

- **Crouch Height Reduction Percentage:** How much (%) the height of the original collider will reduce.
- **Linear Drag:** How much linear drag will increase while crouching, meaning, how much slow down occurs.

Platform Crouch: A crouch that allows the entity to go through a platform, effectively dropping down a platform on a specific time interval.

Platform Crouch Settings:

- **Crouch Height Reduction Percentage:** How much (%) the height of the original collider will reduce.

- **Linear Drag:** How much linear drag will increase while crouching, meaning, how much slow down occurs.
- **Platform Tag:** Tag of the surface that is intended for platform dropping. Note that the layer and tag are different things, and that the surface should still have the intended Ground layer.
- **Platform Hold Time:** How long the input must be held for the action of platform dropping to occur. If the input is lifted before the time, the counter resets.
- **Platform Drop Time:** How long the entity can “phase” through the platform before colliding with it again.

Climb Action

The Climb Action of the VM Module allows for the entity to climb up or down walls or objects with that specific intention. This changes the VM State to Climbing.

General Settings:

- **Climb Up Key:** Selects a key input for climbing up.
- **Climb Down Key:** Selects a key input for climbing down.
- **Climbable Layer:** The layer of which the climbable objects must have to be used.
- **Climb Check Range:** Range of which the entity can check if there is a climbable surface. It’s a line pointing down check based on the origin point of the entity.
- **Climb Speed:** How fast the entity can climb, both up and down.
- **Hold Climb:** By enabling this, the entity will hold the climb by maintaining a static position when the up or down inputs lifted. If not enabled, the entity simply drops the action, effectively falling down the climbable object.

Action/Interaction Category Modules

Action Type Module

The Action Type Module is used to perform an action outside the standards of a platformer. You can use it as an extra action that consumes something, instantiates something, plays an animation or a sound, or even an external action specified by the developer. An action is used when a specific action button is used. You can have an **indefinite** number of these Type Modules per Modular Brain, but only the **first 10** will show up on the editor.

General Settings:

- **Action Input**: Selects the key input for the action.
- **Action Type**: Selects the type of action to be performed.

Instantiate Action

Action that allows the entity to instantiate a prefab, that can be used as a shooting mechanic or object placing mechanic.

Instantiate Action Settings:

- **Prefab:** Game Object that is going to be instantiated, use an object from your prefabs.
- **Position Offset:** Starting position of the instantiated object relative to the entity's position.
- **Angle:** Angle at which the instantiated object is launched. If 0, shoots straightforward, if it moves onto the negatives, it shoots lower, if it moves onto the positives, it shoots higher.
- **Speed:** Speed at which the projectile or object is travelling when instantiated.
- **Use Character Direction:** This enables the instantiation to be entity-based direction, meaning that when the entity flips

directions, the direction of the shooting is also flipped. Without this setting, the entity only shoots to the right.

- **Cooldown:** Time between each action to be able to be performed.
- **Use Mouse to Aim:** Enables mouse-based direction, meaning wherever the mouse is pointing, the instantiated object will be moving in that direction. This overwrites the Angle setting and the Use Character Direction setting.

Consumption Action

Action that actively consumes an item (or not) from the Inventory modules, granting a recovery or a depletion of a resource from the Resource modules.

Consumption Action Settings:

- **Resource Name:** Name of the resource from a Resource Module's resource name.
- **Amount:** How much the resource will be recovered or depleted by doing the action.
- **Action:** Selection of the type of action that affects the resource; can be Recover or Deplete.
- **Cooldown:** Time between each action to be able to be performed.
- **Use An Item:** Toggle if you intend to use an Inventory Item from an Inventory Module.
- **Inventory Module Name:** Name of the module where an item will be consumed from.
- **Item:** Inventory Item object intended for consumption.
- **Item Amount:** How many items will be consumed to perform the action.

Sound Effect Action

Triggers a specific sound that plays on the Audio Source.

Sound Effect Settings:

- **Audio Clip:** Specific audio clip that plays on action.
- **Volume:** Volume for the specific audio clip action.
- **Loop:** Loops the audio clip.
- **Pitch:** Pitch of the specific audio clip action.
- **Cooldown:** Time between each action to be able to be performed.

Animation Action

Triggers a specific action from the Animator.

Animation Settings:

- **Trigger Name**: Name of the animation on the Animator.

External Action

An External Action is an action outside the established action types, entirely defined and coded by the developer.

An External Action is a Scriptable Object. Any script that inherits the External Action class can be used as an External Action. For an example, see "ActionTest" script on the Utils folder and create a SO using "Platform Crafter's Modular System > Utils > External Action Test" menu tab.

Interaction Type Module

The Interaction Module is used when the entity wants to interact with something, for example, an external obstacle that uses the Receptor of the Interaction and does something by pressing the interaction button inside the Area of Interaction of the entity. You can have an **indefinite** number of these Type Modules per Modular Brain, but only the **first 10** will show up on the editor.

General Settings:

- **Automatic Interaction**: This toggles the interaction to be activated automatically and independently from input.
- **Interaction Key**: Selects the key input for the interaction.
- **Interaction Channel**: Object that allows for the communication of the entity and the Interaction Receptor.
- **Interaction Radius**: Radius for the Area of Interaction. The origin point is the entity's origin point.

Interaction Channel

The channel acts as a tunnel communication between the sender (entity) and the receptor of the interaction. Even if multiple interaction object shares the same channel, the interaction will only occur if the interaction was made inside the Area of Interaction.

To create, go to "Platform Crafter's Modular System > Utils > Interaction Channel" tab. Rename it and can be used for an interaction.

Interaction Receptor

The Interaction receptor is a script that can be passed down onto a new script.

Container Category Modules

Resource Type Module

A Resource Module is a container of a certain resource, for example, Health or Mana, used by the entity and can be depleted or recovered. These actions can be done by doing an Action or passively through automatic recovery or depletion. You can have an **indefinite** number of these Type Modules per Modular Brain, but only the **first 10** will show up on the editor.

General Settings:

- **Resource Name:** Name given to the resource. It's best not to leave it empty.
- **Max Value:** Max amount of the resource.
- **Current Value:** Current amount of the resource. This value is changeable by in-game actions, or it can be set to start at any value.
- **Resource Color**: The color of the resource. Can be useful for distinguishing bars and useable on UI.
- **Toggle Passive**: This enables the Passive portion of the Resource Settings.
- **Passive Recovery Rate**: Amount of the Resource to be recovered.
- **Passive Recovery Interval**: Time between each passive recovery.
- **Passive Depletion Rate**: Amount of the Resource to be depleted.
- **Passive Depletion Interval**: Time between each passive depletion.

Inventory Type Module

The Inventory Type refers to the container module capable of having a physical inventory for the storage of items. These items can be stacked, and the inventory can be changed in size. The inventory size is calculated through the grid width and height. You can have an **indefinite** number of these Type Modules per Modular Brain, but only the **first 10** will show up on the editor.

General Settings:

- **Grid Width:** Number of columns of the inventory.
- **Grid Height:** Number of lines of the inventory.
- **Slots**: Number of slots.
- **Initialize Inventory:** Initialize the inventory. If there is a change on the values, all the items are lost.

Inventory Item

The Inventory Modules uses Inventory Items on the slots. These items are SO created on the “Platform Crafter’s Modular System > Utils > Inventory Item” menu tab. Rename this object.

Item Settings:

- **Item Name:** Name of the item. It’s important to use for module interactions.
- **Icon:** Icon that displays for the item on the inventory display of the Inventory Module.
- **Is Stackable:** By stacking it means whenever there’s a new item, it adds it to an already stablished slot with that same item up until its max stack size.
- **Max Stack Size:** Amount of which a slot can fit a certain a quantity of items.

Visuals & Sound Category Modules

Animation Type Module

The Animation Type Module refers to the direct control of animations of the Animator component, each one named after each action pertaining to the HM and VM modules. Each animation play depending on the state of each Physics Module. When a HM and a VM action animation occur at the same time, sometimes you can have fused animations, such as Walk and Crouch, giving you the Crawl animation. You can only have **one** of these Type Modules per Modular Brain.

General Settings:

- **Idle Animation:** Name given to the default state of animation (if the character is not in any HM or VM state).
- **(HM Action) Animation:** Name given to the animation on the animator for a VM action. Note: the name should correspond exactly like name on the Animator.
- **(VM Action) Animation:** Name given to the animation on the animator for a VM action. Note: the name should correspond exactly like name on the Animator.
- **(Fused Action) Animation:** Name given to the animation on the animator for a HM + VM combined action. The combined animations occur when a HM and VM states are active at the same time. The fusions are
  - **Crawl:** (Crouch + Walk) OR (Crouch + Sprint)
  - **Air Dash:** (Jump + Dash) OR (Air Jump + Dash)

Note: the name should correspond exactly like name on the Animator.

The module also has two buttons, one for **Pausing Animations** and another for **Unpausing Animations** on the editor.

Sound Effects Type Module

The Sound Effect Type Module refers to the direct control of sounds that play out of the Sound component, each one named after each action pertaining to the HM and VM modules. When a HM and a VM action sound occur at the same time, sometimes you can have fused sounds, such as Walk and Crouch, giving you the Crawl sound effect. You can only have **one** of these Type Modules per Modular Brain.

General Settings:

- **(Idle, Action, Fused Action) SFX:**
  - **Clip:** The sound clip intended for playing.
  - **Volume:** Volume of the specific SFX.
  - **Loop:** By enabling this, the sound will loop until change of state.
  - **Pitch:** Pitch of the specific SFX.

Note: The logic for the Sound Effect Fused Actions is the same as the Animation module.

The module also has two buttons, one for **Pausing Audio** and another for **Unpausing Audio** on the editor.

Custom Modules

Custom modules refer to a group of modules customized by the developer. Any module outside the already established Type Modules is considered Custom and thus must be used in the Custom section of the Brain.

Currently, there is a custom module on the base files, named **Camera Follow**, which allows for a basic camera following of the entity.

Camera Follow Settings:

- **Smooth Speed:** The speed at which the camera follows the entity. If it is slower, it lags behind the entity, giving it a smooth transition.
- **Offset:** Position offset compared with the current position of the entity.

Note: The **Main Camera** is the only affected camera on the scene, if you wish to change this, simply edit the script.

Others

Shadow Effect Component

The Shadow Effect Component is used separately from the Modular Brain and is used to determine an effect to which a set of shadows or after images appear on the entity. You can use this component outside the Brain; however, it is best used in conjunction with the Modular Brain.

Settings:

- **Shadow:** This is the Prefab object meant to be used as the shadow object. In the Utils folder, there is already a "ShadowObject" ready to be used as the shadow object.
- **Speed:** How fast the shadows are being created.
- **Color:** The base color of the shadows. Remember that the colour should have an alpha value bigger that 0 to be observable.

Solid Component and *ShadowObject*

Used in conjunction with the Shadow Effect component, any shadow object used in the shadow field of the effect should have the Solid component and the Sprite Renderer. If you wish to create a new Shadow Object, be aware of the components it needs.

Solid Settings:

- **Fade Duration:** How much time it takes for a single shadow to fade out and disappear.