EZGamepad and EZGamepadManager (EZGM) v1.0 Documentation

Created by Ronnie Kauanoe

CONTACT: RONNIE.KAUANOE.WORK@GMAIL.COM

Table of Contents

Introduction

The EZGamepad and EZGM were made to simplify Unity's v1.0 Input System and to provide an easy-to-integrate generic gamepad class and manager for single-player and (up to) four-player local multiplayer games.

Features

- Supports up to four-player local multiplayer.
- Supports Xbox One and PlayStation 4 controllers.
  - NOTE 1: Unity recognizes both controllers as generic. Therefore, you can use any combination of the two controllers and this package will still work (e.g. two PS4 controllers, two Xbox One controllers).
  - NOTE 2: Additional drivers needed for later generation (e.g. Xbox 360, PS3) controllers to be detected by Unity.
- EZGamepad class that contains all button and joystick information for a single controller.
  - Example: Want to see if Player One is pressing down the left trigger?

    ```
    EZGamepad ezgp = EZGM.GetEZGamepad(Player.One);
    bool press = ezgp.leftTrigger.isPressed;
    ```

- EZGamepadManager (aka EZGM) class that updates EZGamepads and manages their connections and player assignments **dynamically**.
- Includes a simple example scene and script showcasing the EZGamepad and EZGM's functionality. Can be found at Examples > Sample.
- Includes a visualizer scene. Can be found at Examples > Visualizer. The images on pages 10 and 11 are screenshots of the visualizer in action.

Documentation

EZGamepad: Simplified Gamepad class. Contains button and joystick values.

Constructor	Constructor Description	Parameter Descriptions (if applicable)
EZGamepad()	Empty EZGamepad.	None
EZGamepad(Gamepad gp **)**	Creates EZGamepad using ButtonControls and StickControls from Gamepad class.	Gamepad gp: Gamepad to derive ButtonControls and StickControls from.

Field Name	Type	Description	Xbox One equivalent (if applicable)	PlayStation 4 equivalent (if applicable)
dpadUp	EZButton	Dpad up EZButton.		
dpadDown	EZButton	Dpad down EZButton.		
dpadLeft	EZButton	Dpad left EZButton.		
dpadRight	EZButton	Dpad right EZButton.		
buttonNorth	EZButton	Top face EZButton.	Y	Triangle
buttonEast	EZButton	Right face EZButton.	B	Circle
buttonSouth	EZButton	Bottom face EZButton.	A	Cross
buttonWest	EZButton	Left face EZButton.	X	Square
leftStickPress	EZButton	Left stick press EZButton.	Left Stick Press	L3
rightStickPress	EZButton	Right stick press EZButton.	Right Stick Press	R3
leftShoulder	EZButton	Left shoulder EZButton.	Left Bumper	L1
rightShoulder	EZButton	Right shoulder EZButton.	Right Bumper	R2
leftTrigger	EZButton	Left trigger EZButton.	Left Trigger	L2
rightTrigger	EZButton	Right trigger EZButton.	Right Trigger	R2
startButton	EZButton	Center right EZButton.	Menu	Options

selectButton	EZButton	Center left EZButton	View	Share
ezButtons	EZButton[]	Array of EZButtons. Useful for iterating through all EZButtons.		
leftJoystick	EZJoystick	Left EZJoystick.		
rightJoystick	EZJoystick	Right EZJoystick.		
ezJoysticks	EZJoystick[]	Array of EZJoysticks. Useful for iterating through all EZJoysticks.		
justConnected	bool	Returns true if the EZGamepad was just connected this frame.		
isConnected	bool	Returns true if the EZGamepad is connected.		
justDisconnected	bool	Returns true if the EZGamepad was just disconnected this frame.		

Method Name	Return Type	Method Description	Parameter Descriptions (if applicable)
StartHaptics(float lowFrequency, float highFrequency)	void	Sets the haptics speeds of the left (low frequency) and right (high frequency) motor for this device.	float lowFrequency: Left motor frequency float highFrequency: Right motor frequency.
PauseHaptics()	void	Pauses haptics for this device.	
ResetHaptics()	void	Turns off haptics for this device.	
ResumeHaptics()	void	Resumes haptics on this device. Only resumes if haptics were paused and not reset.	

HapticsOverTime(float duration, float leftFrequency, float highFrequency **)**	void	Starts haptics on this device and stops after a set duration in seconds.	float duration: Time interval the controller will vibrate. float lowFrequency: Left motor frequency. float highFrequency: Right motor frequency.
GetDeviceID()	int	Returns the ID value given to the EZGamepad by Unity.	

EZButton: Simplified ButtonControl class. Contains “just pressed, “is pressed”, and “just released” values and press depth values (if applicable).

Constructor	Constructor Description	Parameter Descriptions (if applicable)
EZButton()	Empty EZButton	
EZButton(ButtonControl bc **)**	Creates EZButton using corresponding ButtonControl.	ButtonControl bc: ButtonControl to derive name, pressed states, and press depth value from.

Field Name	Type	Description
justPressed	bool	Returns true if the EZButton was just pressed this frame.
isPressed	bool	Returns true if the EZButton is currently being held down.
justReleased	bool	Returns true if the EZButton was just released this frame.
pressDepth	float	Press depth of this button this frame. Ranges from 0 to 1 for trigger. Either 0 or 1 for all others.
name	string	Name of the EZButton.

Method Name	Return Type	Method Description	Parameter Descriptions (if applicable)
UpdateButton()	void	Updates this EZButton’s “just pressed”, “is pressed”, “just released”, and press depth values.	

EZJoystick: Simplified StickControl class. Contains X-Axis and Y-Axis values.

Constructor	Constructor Description	Parameter Descriptions (if applicable)
EZJoystick()	Empty EZJoystick.	
EZJoystick(StickControl sc **)**	Create EZJoystick using corresponding StickControl.	StickControl sc: StickControl to derive name and axis values from.

Field Name	Type	Description
xAxis	float	The x-axis value (-1.0 to 1.0) of the EZJoystick.
yAxis	float	The y-axis value (-1.0 to 1.0) of the EZJoystick.
name	string	Name of the EZJoystick.

Method Name	Return Type	Method Description	Parameter Descriptions (if applicable)
UpdateJoystick()	void	Updates this EZJoystick's x- and y-axis values.	

Player: Enumerated identifiers of the players.

NOTE: If Unity allows more than four controllers to be used in their Input System in the future, Player will be updated to reflect the maximum number of possible controllers.

Number	Description
One	The enumerator for Player One.
Two	The enumerator for Player Two.
Three	The enumerator for Player Three.
Four	The enumerator for Player Four.

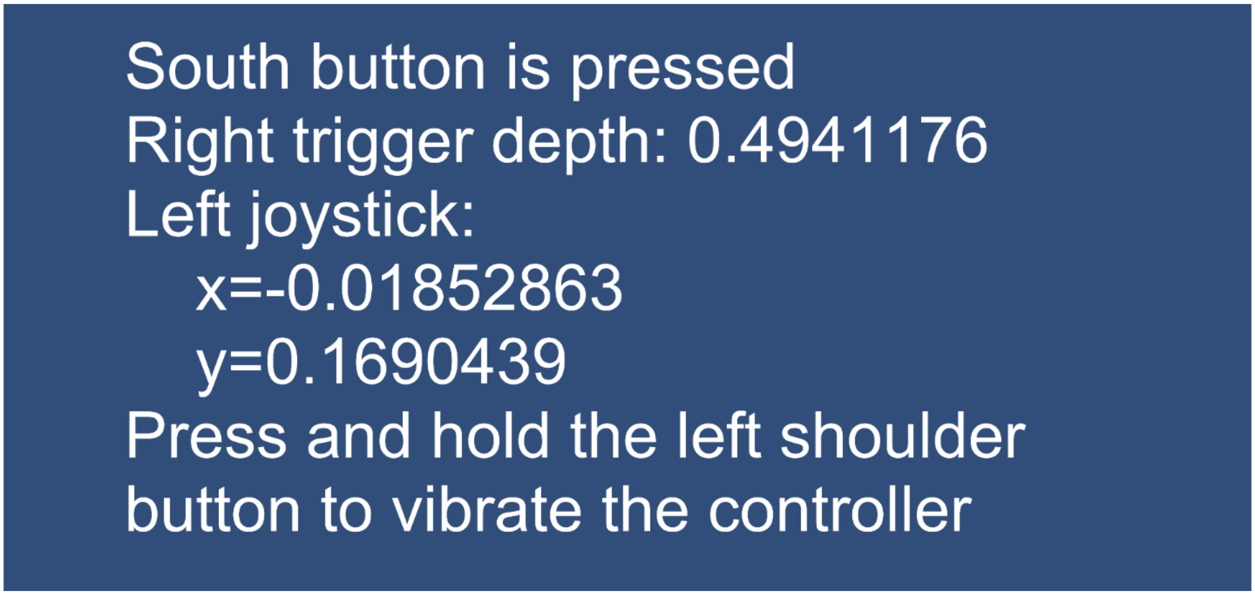
EZGamepadManager (EZGM): Updates and manages the EZGamepads.

NOTE: All methods of EZGM are static and must be called from the class directly. Example: `EZGM.GetEZGamepad(Player.One);`

Field Name	Type	Description
debugConnections	bool	If true, the number of newly added/removed controllers will be displayed.

Method Name	Return Type	Method Description	Parameter Descriptions (if applicable)
GetEZGamepad(Player p **)**	EZGamepad	Returns EZGamepad of specified player. Returns null if the player is not connected.	Player p: Player to retrieve EZGamepad from.
ConfigUpdating()	bool	Returns true if the EZGamepadManager is updating.	
ConfigStoppedUpdating()	bool	Returns true if the EZGamepadManager stopped updating this frame.	
EZGamepadCount()	int	Returns the number of EZGamepads currently connected.	

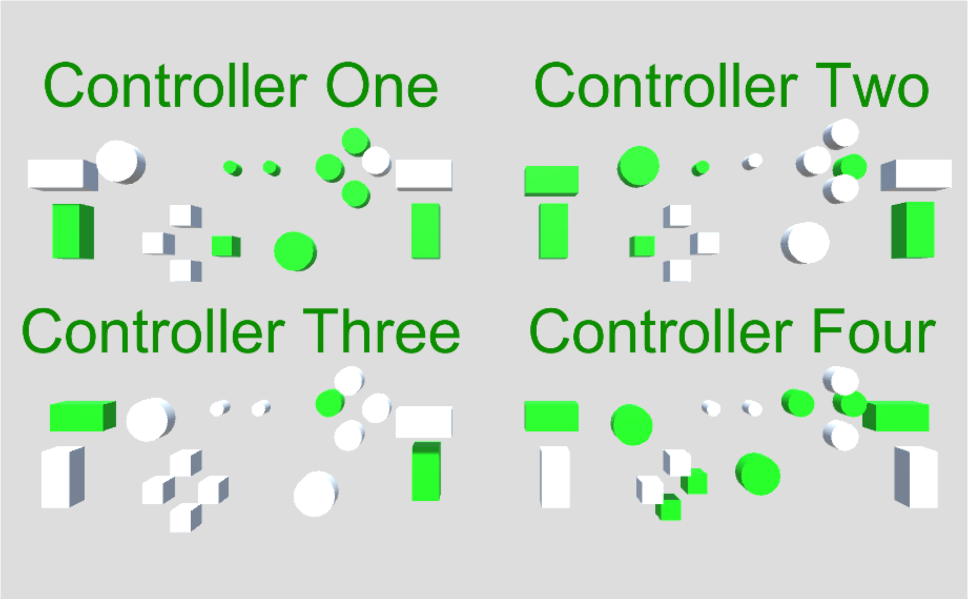
Sample Scene



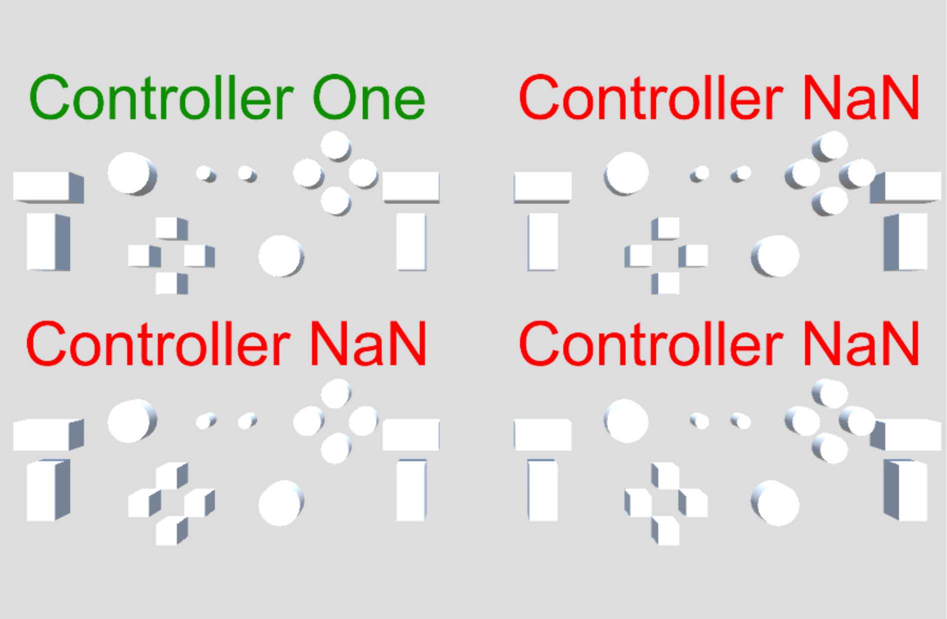
```
South button is pressed
Right trigger depth: 0.4941176
Left joystick:
    x=-0.01852863
    y=0.1690439
Press and hold the left shoulder
button to vibrate the controller
```

The SampleScene (found in Examples > Sample) shows the basic functionality of the EZGamepad and EZGM classes. Firstly, when the status of the south button is displayed and will show “just pressed”, “is pressed”, “just released”, and “not pressed” accordingly. Second, the depth of the right trigger is displayed and will range from 0.0 to 1.0. Third, the x- and y-axes of the left joystick is displayed and will range from -1.0 to 1.0. Finally, if the user holds down the left shoulder, the controller will vibrate.

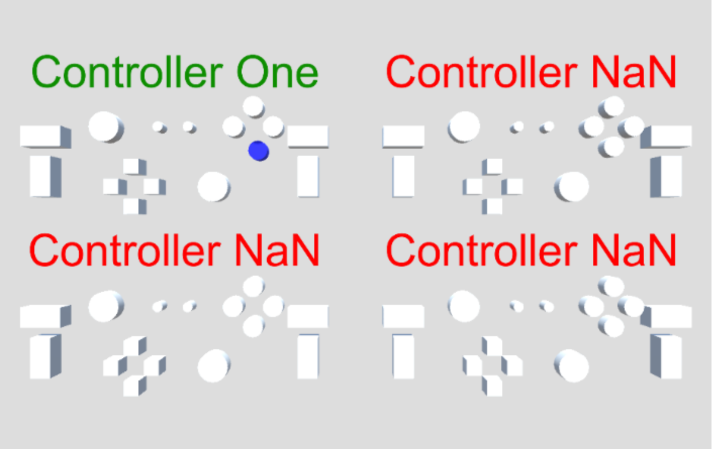
Visualizer Scene



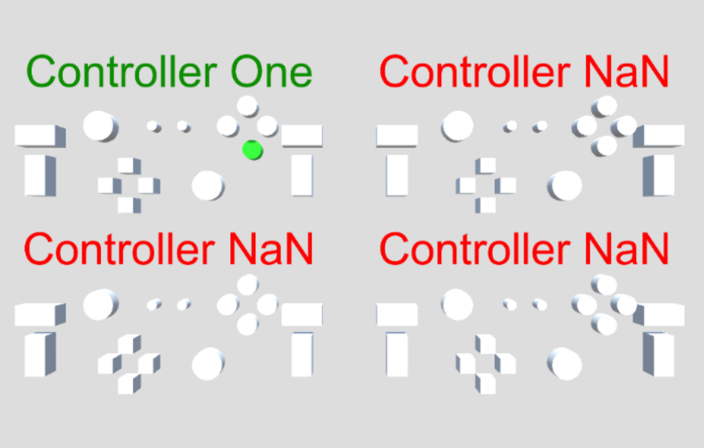
In VisualizerScene (found in Examples > Visualizer), there are four GameObjects representing four controllers, each with a Controller script (found in the "Scripts" folder) on them. If a player is connected, the player's enumerator displayed above the GameObject. Else, the text above the GameObject will say "Controller NaN". Below is an example:



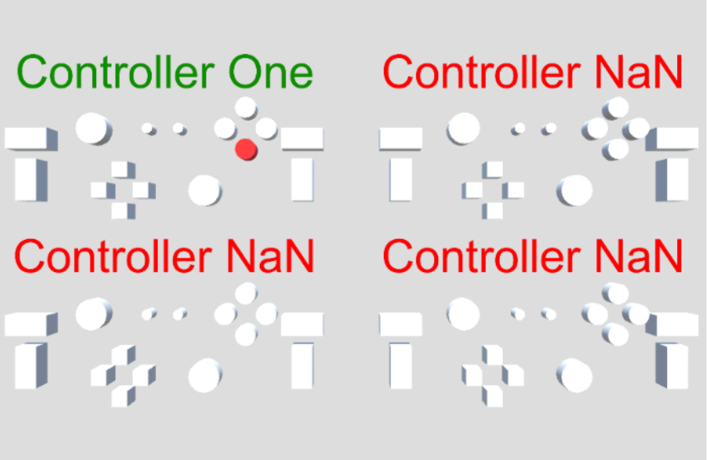
When a button is pressed down, the corresponding GameObject will turn blue for that frame (e.g. south face button).



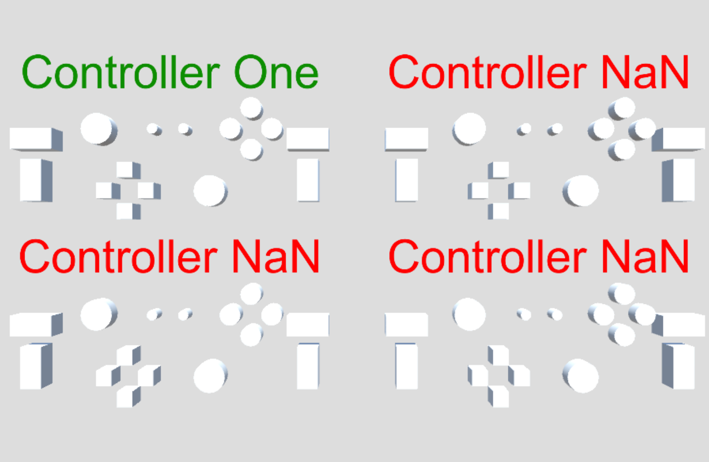
When the button is held down, the object will turn green and remain green until it is let go.



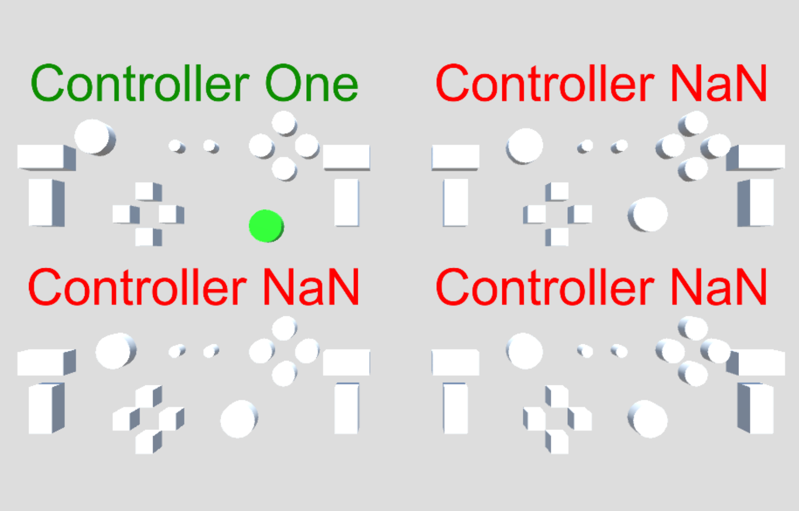
On the frame that it is released, the object will turn red for one frame.



Then, it will go back to its default white color.



Additionally, when you move either of the joysticks, the corresponding joystick GameObject will move in the same manner. Furthermore, the joysticks are also buttons, as demonstrated below.


Controller One
Controller NaN
Controller NaN
Controller NaN