2D Controller - Gamepad

Self-study

Intro to Gamepad Development

If you're using Unity, the Rewired package can save vast amounts of work. Recommended!

Technical

The basic differences where Gamepad is different from Keyboard gaming:

- Analog inputs (up to 2 thumbsticks, with two axes each + additional "axis" on analog L2/R2 (trigger) buttons).
- Nothing to emulate mouse well.
- Less buttons.
- No key ghosting (you can press all the buttons simultaneously and it will transmit).
- Gamepads can vibrate. You can't rely on it, but it can add a lot of feedback to the player.

It is a good idea to add deadzones (i.e. ignore values from e.g. <-0.3,0.3> from the thumbsticks, to ignore accidental presses and small hardware/controller glitches. Also, sometimes a turnback can happen, where the momentum of the thumbstick can "overshoot zero" when released. The deadzone should deal with this too.

Often, libraries/engines will give you "smoothed" input, which slowly transfers from one value to another without "jumps". This is usually beneficial, but in some context you might rather want to utilize "raw" input, which you can (almost) always get too.

Technical meets design

With a gamepad, you need to have the "selected" status visually clear on UI elements. It is a good idea to have that nevertheless.

If you're creating a game with both keyboard **and** gamepad support, you need to consider this:

- It is a good idea to switch between inputs based on what the player is trying to do. Anything on keyboard/mouse was pressed? Switch to keyboard+mouse. Any gamepad key pressed? Switch to gamepad.
- You also should show mouse only when the user is not using gamepad.
- It is good to fallback to keyboard in case of gamepad disconnect (and show mouse, select UI element).

- Change button prompts (don’t say “Press Enter” when the user is using gamepad). Also use the correct button prompt depending on the controller (don’t say press green-A to a user with a PS controller).
    - This can be mitigated if you use visual-context prompts. E.g. Instead of “press green-A” show 4 buttons in the shape and highlight the bottom one.

Design

Gamepads make your game seem more accessible (due to less buttons). The controller schema visualization also looks simpler :).

People not used to gamepads:

- Have difficulties coordinating both thumbsticks.
- Have problems pressing the thumbsticks (R3/L3 buttons), especially combining it with thumbstick movement.

Give your players options to re-map buttons.

The limited amount of buttons on gamepad can be solved by:

- Having actions context specific.
- Holding a button and moving joystick (e.g. radial menus, move+dodge=roll, etc).
- Pressing two buttons simultaneously. (watch out that you need to check this with some delay).
- Holding a button doing something else than pressing it (use sparingly and at least with similar context).

Watch out what is possible to do simultaneously. E.g. actions on d-pad usually make the player stop moving by the left-stick. Shoulder button fingers are not used for anything else, but relatively few people use that claw grip that enables them to press both (e.g. L1 and L2) simultaneously.

If you support multiple players, you have to deal with who has control in common contexts, such as menu. Consider asking other people to confirm, so that someone doesn’t advance menu when others are still looking around.

Unless you need lots of buttons and utilize the d-pad specially, you can let players choose between thumbstick and d-pad movement.

If your game can be controlled by a gamepad, all of the game should be controllable by it (including all menus, etc.)

Joystick is slower and less precise than mouse. Take that into consideration and balance it. If your game is supporting multiplayer between gamepad and keyboard+mouse players, it might be hard to balance that (but sometimes, a good matchmaking algorithm should take care of that by itself).

Sometimes, you have to decide, if the thumbstick is from the direction of the character or the camera. This is especially true, if you use some strange perspective (isometric, external camera alá old Resident Evil).

Gestures:

- The thumbsticks have more flexibility of movement. You can do some motions that make sense with a gamepad only (half-circle then go-to middle, rotating around).

Lab

Start with your previous lab (Controller - Keyboard) result.
Alternatively, in the platformer repo, start with the solution branch

Starting point

scenes/Environment.tscn - A very basic level with platforms.
scenes/Crate.tscn - A crate object to push around.
scripts/ScreenWrap.gd - Screen wrap for the player and the crate (player's script extends the screen wrap).
scenes/Player.tscn - Using RigidBody2D as a base.

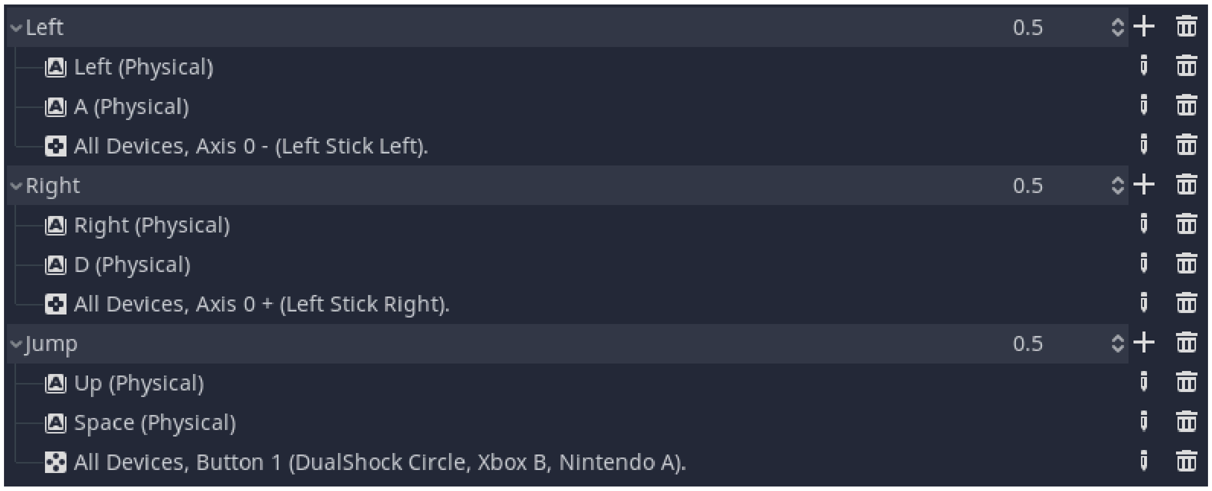
- **scripts/Controller.gd** - Basic controls using keyboard (utilizing InputMap).

Using InputEvent — Godot Engine (stable) documentation in English
Input examples — Godot Engine (stable) documentation in English
RigidBody2D — Godot Engine (stable) documentation in English
Godot 3.0: Rigid Bodies · KCC Blog (kidscancode.org) (setting position of RigidBody2D)
Physics introduction — Godot Engine (stable) documentation in English

Steps:

InputMap setup

Project → Project Settings… → Input Map tab
Add axes mapping.



Left 0.5
Left (Physical)
A (Physical)
All Devices, Axis 0 - (Left Stick Left).
Right 0.5
Right (Physical)
D (Physical)
All Devices, Axis 0 + (Left Stick Right).
Jump 0.5
Up (Physical)
Space (Physical)
All Devices, Button 1 (DualShock Circle, Xbox B, Nintendo A).

Support of analog input

You need to support analog input. There are two functions for this:
`Input.get_action_strength(axis)` and
`Input.get_action_raw_strength(axis)`. Called like this:
`Input.get_action_strength("Left")`

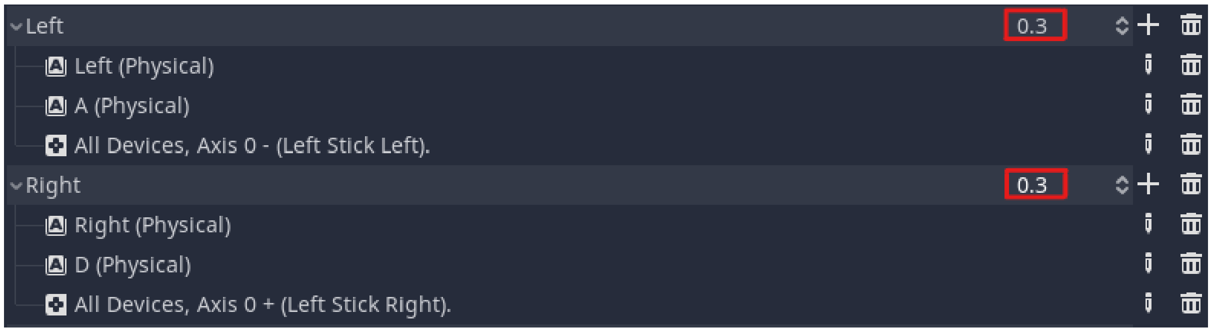
Modify files:
scripts/Controller.gd - Analog reading, `Input.get_action_strength()`.
[Input — Godot Engine (stable) documentation in English]

Working with axis

You can get opposite axes at once:
`Input.get_axis("Left", "Right")`
scripts/Controller.gd - Getting both directions at once using `Input.get_axis()`.
[Input — Godot Engine (stable) documentation in English]

Dead zone setup in InputMap

Project → Project Settings… → Input Map tab



Left 0.3
Left (Physical)
A (Physical)
All Devices, Axis 0 - (Left Stick Left).
Right 0.3
Right (Physical)
D (Physical)
All Devices, Axis 0 + (Left Stick Right).

Dead zone as a parameter in the Inspector

Try to set up your own dead-zone (where the controller won't react) in your custom code.
`Input.get_action_raw_strength("Left")`
scripts/Controller.gd - Export a variable for the dead zone percentage, then threshold the raw strength of the action.
Input — Godot Engine (stable) documentation in English

Vibration on collision with ground

Whenever the player collides with the ground, the gamepad vibrates.
e.g. `Input.start_joy_vibration(0, 0.2, 0.3, 0.1)`
scripts/Controller.gd - Export variables for the vibration parameters and then use them.
Input — Godot Engine (stable) documentation in English

Useful resources

Controllers, gamepads, and joysticks — Godot Engine (stable) documentation in English
Input — Godot Engine (stable) documentation in English

Assets sources

Kenney • Background Elements Redux
Kenney • Platformer Pack Redux
Kenney • Platformer Characters
Cannon Ball Assets – RedFoc